MASTER EN SCIENCES MATHÉMATIQUES

Deux méthodes de résolution par linéarisation d'un problème non linéaire en variables mixtes

Warnant, Dominique

Award date:
1993

Awarding institution:
Universite de Namur

"Deux méthodes de résolution par linéarisation d'un problème non-linéaire en variables mixtes"

Dominique Warnant
Juin 1993

Facultés Universitaires Notre-Dame de la Paix
FACULTE DES SCIENCES
Rue de Bruxelles 61 - 5000 NAMUR
Tél. 081/72.41.11 - Télex 59222 facnam-b - Téléfax 081/23.03.91

"Deux méthodes de résolution par linéarisation d'un problème non-linéaire en variables mixtes"

WARNANT Dominique

Résumé

Deux algorithmes permettant de résoudre deux cas particuliers de programmes non-linéaires mixtes sont décrits dans ce mémoire. Chacune des deux méthodes proposées est basée sur la résolution d'une suite d'approximations linéaires du problème de départ.

Dans la première méthode, on construit itérativement un modèle linéaire par morceaux de la fonction objectif, restreinte à un ensemble discret Y. La minimisation de ce modèle fournit le prochain point itératif.

Dans la seconde méthode, on linéarise la fonction objectif f et la contrainte g en un point, à chaque itération. Dans certains cas bien précis, on améliore le modèle en linéarisant f et g en plusieurs points.

La convergence des deux algorithmes proposés est démontrée.

Mémoire de licence en Sciences Mathématiques
Juin 1993
Promoteur : Prof. J.J. STRODIOT

Je tiens à exprimer mes plus vifs remerciements au Professeur J.J. Strodiot pour l'intérêt qu'il a porté à ce travail, pour sa disponibilité et ses nombreux conseils.

Je voudrais également témoigner ma reconnaissance à mes parents pour m'avoir permis d'entreprendre les études que j'avais choisies.

Enfin, un grand merci à toute ma famille, mes amis et à Johan pour m'avoir soutenue et guidée tout au long de ces quatre années.

Table des matières

Introduction

0.1. Problèmes abordés :

Dans ce travail, on cherche à résoudre le problème en variables mixtes suivant:

$$(P) \qquad \begin{aligned} \min\ & f(x,y) \\ s.c.\ \ & g(x,y) \leq 0 \\ & x \in X \\ & y \in Y \end{aligned}$$

où f et g sont des fonctions de $\mathbb{R}^n \times \mathbb{R}^m$ dans $\mathbb{R}$, X est une partie de $\mathbb{R}^n$ et Y est une partie discrète, finie de $\mathbb{R}^m$.

Le problème est dit "en variables mixtes" car certaines variables du problème (ici, les variables y_i, i = 1..m) ne peuvent prendre que des valeurs discrètes, tandis que les variables x peuvent prendre des valeurs réelles continues. Si toutes les variables étaient contraintes à des valeurs discrètes, on dirait que (P) est un problème " en variables discrètes".

Les deux méthodes développées dans ce mémoire consistent à résoudre une suite d'approximations linéaires du problème de départ.

Le **premier chapitre** est principalement basé sur [6]. Le type de problèmes concernés dans cette section est un cas particulier du problème (P) et peut se formuler comme suit:

$$(P_1) \qquad \min \ f(y) \quad \text{s.c.} \quad y \in B^m = \{0,1\}^m$$

où f est une fonction de $\mathbb{R}^m$ dans $\mathbb{R}$, différentiable et strictement convexe.

Dans l'approche décrite dans ce premier chapitre, on utilise la notion de sous-gradient de la fonction f restreinte à $\{0,1\}^m$ pour construire un modèle linéaire par morceaux de la fonction objectif, et pour établir une condition nécessaire et suffisante d'optimalité:

Une solution admissible du problème (P) est optimale

ssi

un sous-gradient de la fonction objectif restreinte à $\{0,1\}^m$ en cette solution vaut 0.

L'algorithme du sous-gradient recherche la solution itérativement parmi des points admissibles. A chaque itération, il génère le point itératif suivant en résolvant le problème pour un modèle linéaire par morceaux qui est construit avec des hyperplans d'appui de la fonction objectif restreinte à $\{0,1\}^m$, hyperplans définis aux points itératifs déjà générés. Une façon de déterminer ces hyperplans d'appui est détaillée dans ce premier chapitre, ainsi qu'une façon de résoudre, à chaque itération, la minimisation du modèle linéaire par morceaux.

Pour terminer cette section, on propose une implémentation parallèle de l'algorithme du sous-gradient.

Le **deuxième chapitre** de ce travail est inspiré de [3]. Il concerne le problème

$$(P) \qquad \begin{aligned} \min\ & f(x,y) \\ s.c.\ \ & g(x,y) \le 0 \\ & x \in X \\ & y \in Y \end{aligned}$$

où f et g sont des fonctions de $\mathbb{R}^n \times \mathbb{R}^m$ dans $\mathbb{R}$, X est une partie de $\mathbb{R}^n$ et Y est une partie discrète, finie de $\mathbb{R}^m$.

La méthode s'appuie sur l'équivalence du problème (P) et du problème suivant:

$$(P_0) \qquad \begin{aligned} \min_{y}\ & F(y) \\ s.c.\ \ & y \in Y \cap V \end{aligned}$$

où $V = \{ y \in Y \mid \exists\, x \in X \text{ t.q. } g(x,y) \le 0 \}$ et F(y) est la valeur optimale du problème

$$P(y) \qquad \begin{aligned} \min_{x}\ & f(x,y) \\ s.c.\ \ & g(x,y) \le 0 \\ & x \in X \end{aligned}$$

On décompose donc le problème, en considérant d'abord le problème par rapport à $y \in Y$. En effet, la méthode consiste en une procédure itérative où on détermine d'abord un candidat discret $\bar{y}$, par la résolution d'une linéarisation de (P). Ensuite, on cherche, si c'est possible, un point

$\bar{x}$ correspondant, via la résolution du problème P($\bar{y}$). Suivant les cas, on va soit accepter ($\bar{x}$,$\bar{y}$) comme nouvel itéré, soit le rejeter et modifier la linéarisation du problème (P) afin de trouver d'autres candidats. Les modifications possibles sont une linéarisation des fonctions f et g par morceaux, l'introduction d'une région de confiance,...

Plusieurs tests d'arrêt sont mis au point et la convergence de l'algorithme est démontrée, en supposant f et g convexes.

Avant d'aborder ces deux méthodes, il nous faut encore préciser certaines notions de base, utilisées tout au long de ce travail. C'est l'objet de la deuxième partie de cette introduction.

Enfin, la dernière partie de celle-ci donne quelques arguments visant à montrer la difficulté de l'optimisation en variables discrètes (ou mixtes) par rapport aux problèmes en variables continues.

0.2. Notions de base :

Rappelons-nous d'abord la forme générale des problèmes abordés:

$$(P) \qquad \begin{aligned} \min\ & f(x,y) \\ s.c.\ & g(x,y) \leq 0 \\ & x \in X \\ & y \in Y \end{aligned}$$

où $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$

$g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$

$X \subset \mathbb{R}^n$

$Y \subset \mathbb{R}^m$, discret, fini

Définition 1:

Un point $(\bar{x},\bar{y})$ est *admissible* pour (P)

si $g(\bar{x},\bar{y}) \leq 0$

$\bar{x} \in X$

$\bar{y} \in Y$

Définition 2:

Le point (x^*,y^*) est appelé *minimiseur global* pour le problème (P)

si 1) (x^*,y^*) est admissible pour (P)

2) $f(x^*,y^*) \leq f(x,y) \quad \forall (x,y)$ admissible pour (P)

Dans ce cas, la valeur $f(x^*,y^*)$ est appelée *minimum global*.

figure 1:

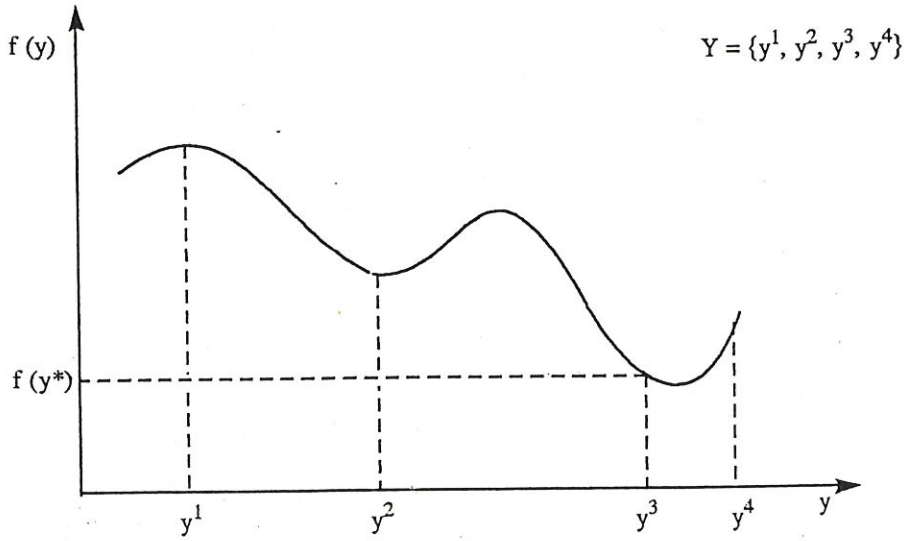


Figure 1. : Illustration de la définition 2. Le minimiseur global de la fonction f (y) sur Y est le point $y^* = y^3$; le minimum global est f (y*).

Définition 3:

Le *pas minimum de la discrétisation* associé à l'ensemble discret Y est la plus petite distance entre deux points distincts de Y, c'est à dire le nombre noté ρ_m défini par

$$\rho_m = \min \{ \, \| z - y \|_\infty \mid y,z \in Y \}$$

Exemple:

Si $Y = \{1, 2.5 , 6.8\}$, alors le pas minimum de discrétisation associé à Y est le nombre ρ_m égal à 1.5

Définition 4:

Un point $u^* = (x^*,y^*) \in X \times Y$ est un *point admissible isolé* de (P)

si 1) u^* est admissible pour (P)

2) $\exists\ \rho \geq \rho_m$ t.q. $\forall\ u = (x,y) \in X \times Y$ t.q. $0 < \|u - u^*\|_\infty \leq \rho$

u n'est pas admissible pour (P)

Remarque :

Autrement dit, à une distance inférieure à ρ de ce point, il n'existe pas de points admissibles pour (P).

Définition 5:

Un point $u^* = (x^*,y^*) \in X \times Y$ est un *minimiseur local* de (P)

si $\exists\ \rho \geq \rho_m$ vérifiant les conditions suivantes:

1) $\exists\ u = (x,y)$ admissible pour (P) t.q. $0 < \|u - u^*\|_\infty \leq \rho$

2) $\forall\ u = (x,y)$ admissible pour (P) t.q. $0 < \|u - u^*\|_\infty \leq \rho$

on a $f(x^*,y^*) \leq f(x,y)$

Remarque :

La condition (1) revient à dire que le point (x^*,y^*) n'est pas un point isolé.

figure 2:

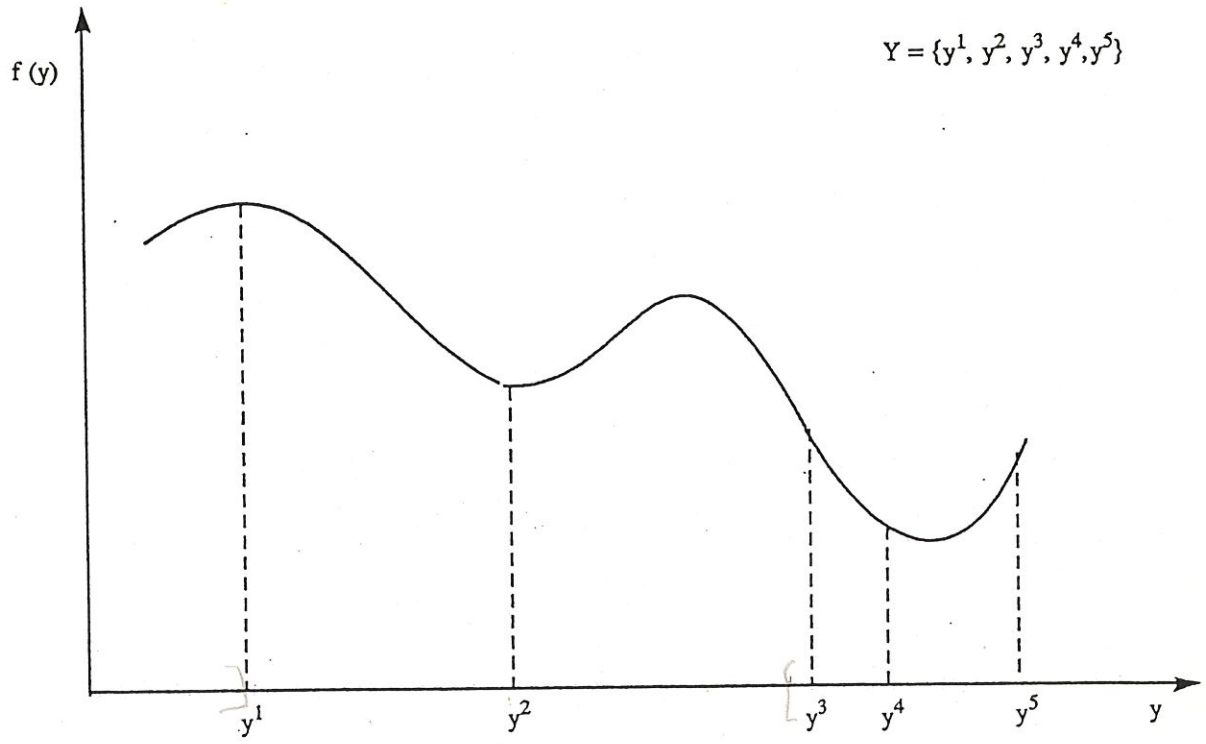


Figure 2. : Illustration de la définition 5. Les points y^2 et y^4 sont deux minimiseurs locaux.

0.3. Difficulté des problèmes d'optimisation en variables discrètes (ou mixtes) :

Il y a plusieurs raisons pour lesquelles les problèmes en variables discrètes (ou mixtes) sont plus compliqués à résoudre que leurs homologues continus (c'est-à-dire les problèmes où les contraintes discrètes n'existent pas):

1) La prolifération des minima locaux se présente même si le problème continu sous-jacent, obtenu en relâchant les contraintes discrètes, est convexe.

exemple:

Considérons le problème (P):

$$\min \; -x_1 - 1.8\, x_2$$

$$\text{s.c.} \quad x_1^2 + (x_2+6)^2 - 85 \leq 0$$

$$1 - x_1^2 \leq 0$$

$$- x_2 \leq 0$$

$$x_1, x_2 \text{ entiers}$$

figure 3:

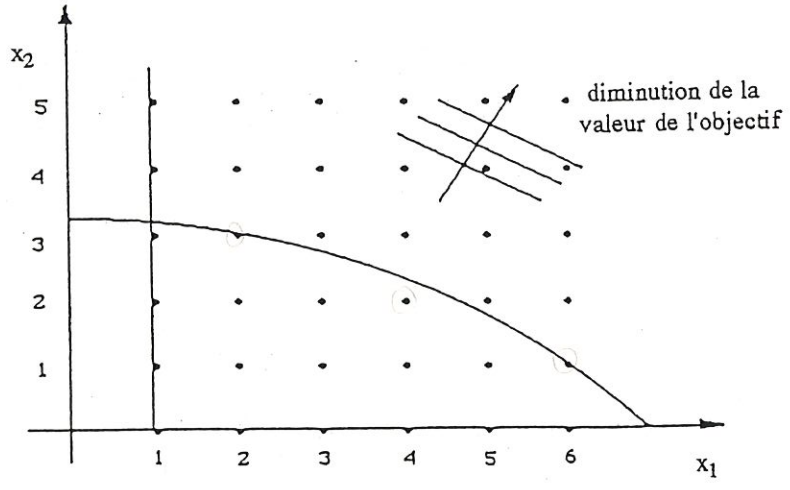


Figure 3.: **Un problème convexe continu avec trois minimiseurs locaux**

Le problème continu (où $x_1, x_2 \in \mathbb{R}$) est convexe et il a un et un seul minimiseur global (et local) qui est $(4.477, 2.059)^T$. Cependant, le problème discret possède 3 minimiseurs locaux qui sont $(2,3)^T$, $(4,2)^T$ et $(6,1)^T$, comme le montre la figure 3. Le minimiseur global discret est le point $(6,1)^T$.

2) Jusqu'à présent, il n'existe pas de critère d'optimalité tel que celui de Kuhn-Tucker. Cela implique que quand la procédure de calcul se termine, il est difficile de garantir que le point trouvé soit un minimum global, à moins qu'une énumération implicite ou explicite de tout autre point n'ait été considérée.

3) Il se peut que le minimiseur global discret soit " très éloigné " du minimiseur global du problème continu. Donc, des procédures qui déterminent la solution du problème relâché puis qui cherchent une solution discrète dans son voisinage ne trouvent souvent qu'un minimiseur local, ou même aucune solution discrète admissible.

De plus, un autre problème se pose car les procédures de résolution des programmes continus ne garantissent en général que de trouver un minimiseur local et pas toujours un minimiseur global! La méthode décrite ci-dessus (problème relâché puis recherche dans le voisinage) a donc encore moins de probabilité de trouver un minimiseur global discret!

exemple:

Soit le problème:

$$\min \; f(x_1,x_2) = (x_1 - 3)^2 + (x_2 - 4)^2$$

$$\text{s.c.} \quad x_1 + 3x_2 - 7.5 \leq 0$$

$$x_1 - 3 \leq 0$$

$$- x_1 \leq 0$$

$$- x_2 \leq 0$$

$$x_1, x_2 \text{ entiers}$$

figure 4:

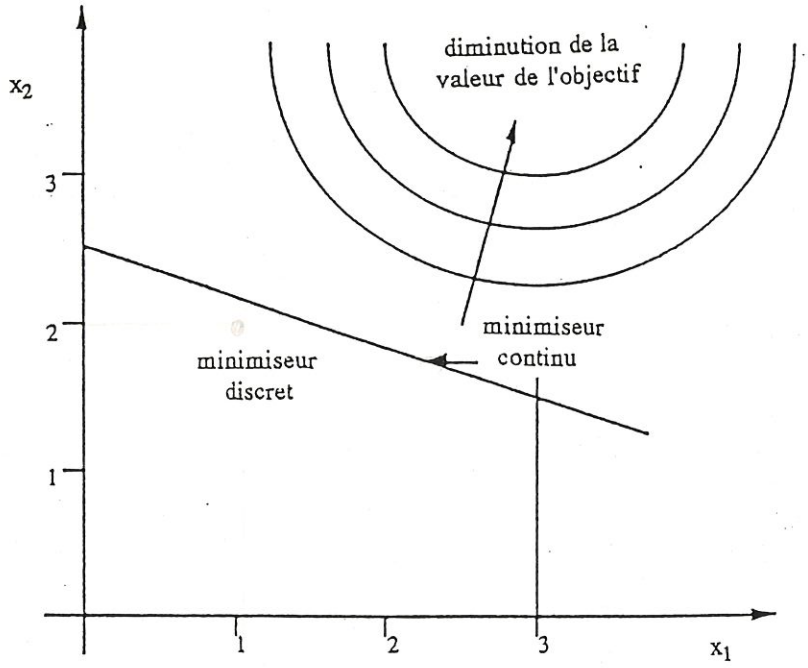


Figure 4.: **Le minimiseur discret n'est pas dans le voisinage discret du minimiseur continu**

Le minimiseur du problème relâché est $(2.251, 1.750)^T$ tandis que le minimiseur global discret est $(1,2)^T$. Une recherche locale autour du point $(2.251, 1.750)^T$ aurait donné comme solution le point $(3,1)^T$, pourtant on voit bien que $f(3,1) = 9$, alors que $f(1,2) = 8$.

Ces difficultés impliquent que les techniques de résolution de problèmes d'optimisation en variables discrètes (ou mixtes) sont généralement coûteuses au point de vue du temps calcul. Par exemple, des techniques telles que " Branch and Bound " [1] qui garantissent de trouver un minimum global, nécessitent beaucoup d'évaluations de fonctions et deviennent quasi-irréalisables

pour des problèmes de grande taille ou des problèmes où les évaluations de fonctions sont particulièrement " chères ".

Il existe bien sûr plusieurs méthodes de résolution de problèmes mixtes, linéaires ou non, mais il serait utopique et hors de propos de vouloir les décrire toutes ici, ne fut-ce que sommairement.

Nous nous limiterons donc à la description et étude de convergence de deux algorithmes se basant sur une linéarisation du programme non-linéaire de départ.

Chapitre 1: Algorithme du sous-gradient pour un programme non-linéaire discret sans contraintes

1.1. Introduction:

Dans ce chapitre, on s'intéresse à la résolution d'une classe de problèmes non-linéaires discrets sans contraintes de la forme suivante:

$$(P_1) \qquad \min f(y) \quad s.c. \quad y \in Y$$

où f est une fonction de $\mathbb{R}^m$ dans $\mathbb{R}$ et Y est une partie discrète, finie de $\mathbb{R}^m$.

Remarquons dès à présent que la restriction de f à l'ensemble discret Y sera notée

$$f^r : Y \to \mathbb{R}$$

On appellera f la fonction objectif continue, et f^r la fonction objectif discrète.

La notion de sous-gradient de la fonction objectif sera définie dans le paragraphe 1.2. . Grâce à cela, on va construire une condition nécessaire et suffisante d'optimalité. On développera un nouvel algorithme, appelé algorithme du sous-gradient. Celui-ci consiste en une procédure itérative qui, à chaque itération, génère le point suivant à considérer en opérant la minimisation sur un modèle linéaire par morceaux de la fonction objectif de départ. Ce modèle sera construit via des hyperplans d'appui à la fonction objectif, définis à chaque étape au point courant. Tout cela sera décrit dans les paragraphes 1.3. et 1.4.

Le paragraphe 1.5. sera consacré à la preuve du fait que l'algorithme se termine et converge vers la solution globale du problème (P1) de départ.

La résolution des sous-problèmes de minimisation des modèles linéaires par morceaux se fera en résolvant un programme linéaire discret équivalent, par des méthodes "standard" de programmation linéaire discrète. On décrira brièvement au paragraphe 1.6. une méthode possible.

Enfin, afin de calculer de " bons " hyperplans d'appui, on va se limiter à un domaine admissible de la forme " $y \in B^m=\{0,1\}^m$ ". Ce calcul est très important à la fois pour la construction du modèle linéaire par morceaux de la fonction f, et pour le test d'optimalité. Il fera l'objet du paragraphe 1.7.

Pour terminer ce chapitre, nous décrirons brièvement l'implémentation parallèle de l'algorithme du sous-gradient. Cela sera le sujet de la section 1.8.

1.2. Notions utilisées :

Certains aspects de l'approche que nous allons décrire, tels que les notions de sous-gradient et hyperplans d'appui, n'ont pas encore été explicitement définis. Ce paragraphe va permettre de nous fixer les idées à ce propos.

1.2.1. Définitions :

Considérons f^r, la restriction de f à Y.

La fonction f^r n'est définie que sur l'ensemble discret $Y \subset \mathbb{R}^m$ et donc f^r n'est pas différentiable. Les notions de gradient et d'hyperplan tangent (notions valables pour des fonctions différentiables uniquement) vont cependant pouvoir être généralisées à notre fonction, grâce aux "sous-gradients" et aux " hyperplans d'appui ", dont voici les définitions.

Définition 1.2.1.1. :

Un vecteur $s \in \mathbb{R}^m$ est un *sous-gradient* de f^r en $\bar{y} \in Y$

si

$$f^r(y) \geq f^r(\bar{y}) + s^T (y - \bar{y}) \quad \forall\, y \in Y$$

Commentaire:

A deux dimensions, cela se traduit souvent en disant que s est un sous-gradient de f^r en $\bar{y}$ si la droite de direction s passant par $\bar{y}$ est sous le graphe de f^r.

Définition 1.2.1.2. :

Le sous-différentiel de f^r en $\bar{y}$ ($\bar{y} \in Y$) est l'ensemble de tous les sous-gradients de f^r en $\bar{y}$,

c-à-d

$$\delta f^r(\bar{y}) = \{ s \in \mathbb{R}^m \mid f^r(y) \geq f^r(\bar{y}) + s^T (y - \bar{y}) \ \forall\ y \in Y \}$$

Définition 1.2.1.3.:

Un *hyperplan d'appui* de f^r en $\bar{y} \in Y$ est un hyperplan défini par

$$g(y) = f^r(\bar{y}) + s^T (y - \bar{y}) \quad \text{où } s \in \delta f^r(\bar{y})$$

Commentaire :

Un hyperplan d'appui de f^r en $\bar{y} \in Y$ est donc un hyperplan qui se trouve, en tous points de Y, sous le graphe de f^r.

figure 1.1.:

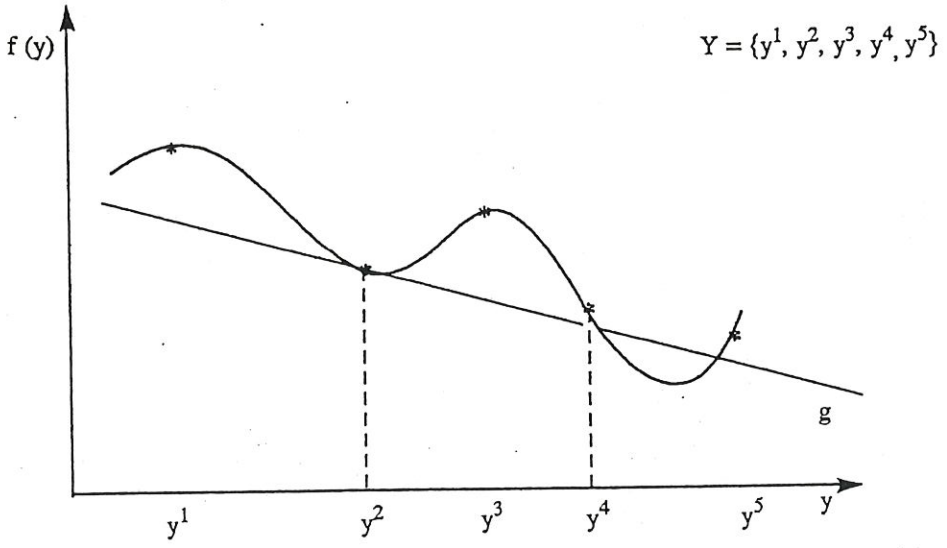


Figure 1.1. : Illustration de la définition 1.2.1.3.; g est un hyperplan d'appui de f^r en $\bar{y} = y^2$ appartenant à Y.

● Remarquons qu'étant donné un sous-gradient, on peut définir un hyperplan d'appui et vice-versa. Donc, les notions de sous-gradient et d'hyperplan d'appui sont fortement liées!

● Au paragraphe 1.7., on va définir une procédure de relèvement pour trouver "un meilleur" hyperplan d'appui de f^r sur $B^m = \{0,1\}^m$, c'est-à-dire un hyperplan d'appui qui " serre " le graphe de f^r le plus étroitement possible. Cette classification se comprend relativement bien dans l'exemple suivant: on considère 3 hyperplans d'appui de la fonction f^r définie sur {0,1}. L'hyperplan B est "meilleur" que A, et l'hyperplan C est " le meilleur ". (cf figure 1.2.)

figure 1.2.

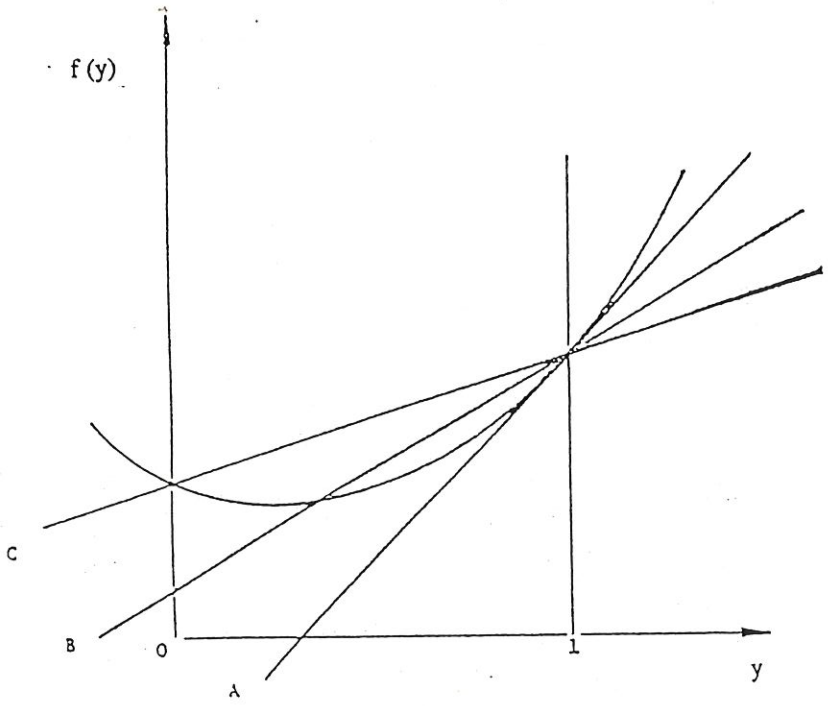


Figure 1.2.: Trois exemples simples d'hyperplans d'appui

1.2.2. Théorème 1.2.2.:

Soit f: $\mathbb{R}^m \rightarrow \mathbb{R}$ convexe, différentiable

Soit $\nabla f(\bar{y})$, le gradient de f en $\bar{y}$

Alors $\nabla f(\bar{y}) \in \delta f^r(\bar{y}) \quad \forall\ \bar{y} \in Y$

preuve:

Pour prouver la thèse, il suffit de montrer que

$\forall\ \bar{y} \in Y$: $\nabla f(\bar{y})^T\ (y - \bar{y}) \leq f(y) - f(\bar{y}) \qquad \forall\ y \in Y$

Si $y = \bar{y}$, c'est évident.

Supposons que $y \neq \bar{y}$, $y \in Y$.

Comme f est différentiable, la dérivée directionnelle de f en $\bar{y}$ dans la direction $(y - \bar{y})$ est définie comme

$$\lim_{\lambda \searrow 0} \frac{f(\bar{y} - \lambda (y - \bar{y})) - f(\bar{y})}{\lambda}$$

On peut montrer que cette limite existe et vaut

$$\nabla f(\bar{y})^T (y - \bar{y}) \qquad (1)$$

D'autre part, comme f est convexe par hypothèse, on a , $\forall \lambda \in]0,1[$

$$f(y) - f(\bar{y}) = \frac{\lambda f(y) + (1-\lambda) f(\bar{y}) - f(\bar{y})}{\lambda}$$

$$\geq \frac{f(\lambda y + (1-\lambda)\bar{y}) - f(\bar{y})}{\lambda}$$

$$= \frac{f(\bar{y} + \lambda (y - \bar{y})) - f(\bar{y})}{\lambda}$$

Et donc

$$f(y) - f(\bar{y}) \geq \lim_{\lambda \searrow 0} \frac{f(\bar{y} + \lambda (y - \bar{y})) - f(\bar{y})}{\lambda}$$

$$\overset{(1)}{=} \nabla f(\bar{y})^T (y - \bar{y})$$

■

Conséquence :

Si au départ, on a une fonction f: $\mathbb{R}^m \rightarrow \mathbb{R}$, convexe, différentiable, et qu'on la restreint à un domaine discret Y, alors, $\forall\ \bar{y} \in Y$, on a directement un sous-gradient (et donc un hyperplan d'appui) qui vaut $\nabla f(\bar{y})$. Cependant, on verra par la suite que dans le cas où $Y = B^m$, il y a moyen de trouver un " meilleur " sous-gradient que $\nabla f(\bar{y})$.

1.3. Une condition nécessaire et suffisante d'optimalité :

Théorème 1.3.1. :

Soit y^* un élément de l'ensemble discret Y,

y^* minimise f^r sur Y

$$\Updownarrow$$

$$0 \in \delta f^r(y^*)$$

preuve :

Par définition du sous-gradient,

$$0 \in \delta f^r(y^*) \quad \Leftrightarrow \quad f^r(y^*) \leq f^r(y) - 0\,(y - y^*) \qquad \forall\, y \in Y$$

$$\Leftrightarrow \quad f^r(y^*) \leq f^r(y) \qquad \forall\, y \in Y$$

■

Remarque :

Le sous-gradient d'une fonction en un point peut ne pas être unique! Il peut y en avoir une infinité. Or, il n'existe pas de méthode générale pour calculer $\delta f^r(y)$ tout entier, d'où la difficulté de vérifier cette condition nécessaire et suffisante.

C'est pourquoi, dans les sections suivantes, on va développer d'autres techniques ne nécessitant pas de connaître le sous-différentiel tout entier.

1.4. Algorithme du sous-gradient :

Nous venons de décrire une condition nécessaire et suffisante d'optimalité, qui sera un critère d'arrêt de notre algorithme. Ce dernier comprend également un deuxième test d'arrêt (celui-ci sera justifié au théorème 1.5.1 de la section suivante). Ce test est le suivant: *si le nouveau point généré a déjà été considéré dans une itération précédente, on est à la solution optimale.* L'algorithme peut donc s'arrêter.

Ce deuxième critère permet d'éviter tout cyclage et il garantit que l'algorithme se termine en un nombre fini d'étapes (la démonstration formelle de ce dernier point est faite dans le paragraphe suivant).

1.4.1. Déroulement de l'itération i de l'algorithme :

L'algorithme consiste en une procédure itérative qui agit à partir d'un point initial y^0 de la façon suivante :

Soit y^i le point courant à la $i^{ème}$ itération.
Au début de la $i^{ème}$ itération:
si $y^i = y^j$ pour un certain $j < i$
ou
si f^r a un sous-gradient nul en y^i
alors,
l'algorithme stoppe et y^i est la solution optimale
sinon
un hyperplan d'appui g_i de la fonction objectif en y^i est généré. L'ensemble de tous les hyperplans d'appui générés (g_j, j = 1..i) définit un modèle linéaire par morceaux de la fonction objectif, qui a la forme suivante:

$$p^i(x) = \max_{1 \le j \le i} \{ g_j(x) \}$$

A cet endroit de la procédure, l'algorithme résout le problème discret pour le modèle linéaire par morceaux. Il prend la solution optimale du problème linéarisé comme prochain point d'itération y^{i+1} et passe à l'itération suivante.

1.4.2. Algorithme formalisé :

0. INITIALISATION :

$T = \emptyset$ (T = ensemble des points déjà considérés)

$H = \emptyset$ (H = ensemble des hyperplans d'appui)

$i = 0$

Choisir $y^i \in Y$

1. ITERATION :

Tant que $i \leq m$ (où m = borne limite du nombre d'itérations)

♦ test d'optimalité :

Si $y^i \in T$ ou si $0 \in \delta f^r(y^i)$

alors y^i est solution optimale.

♦ construction des hyperplans d'appui :

$T = T \cup \{ y^i \}$

$H = H \cup \{ g_i : g_i(y) = f^r(y^i) + (s^i)^T (y - y^i)$ où $s^i \in \delta f^r(y^i) \}$

♦ résolution d'un problème minimax linéaire discret :

Trouver une solution y* du problème

$$\min_{y \in Y} \{p(y) = \max\{g(y), g \in H\}\}$$

♦ mise à jour :

$i = i + 1$

$y^i = y^*$

Fin du "tant que"

Remarque :

On décrira dans le paragraphe 1.7. la façon de choisir le sous-gradient s_j lors de la construction des hyperplans d'appui.

1.5. Propriétés de convergence et terminaison :

Comme on l'a vu précédemment, pour trouver la solution du problème non-linéaire discret original, l'algorithme résout une séquence de sous-problèmes linéaires discrets. Chacun d'eux correspond à un problème de minimisation discrète pour une fonction linéaire par morceaux, définie par un groupe d'hyperplans d'appui de la fonction objectif.

Nous noterons p^i et p^{i+1} les fonctions linéaires par morceaux générées respectivement à la $i^{ème}$ et à la $(i+1)^{ème}$ itération.

La fonction p^{i+1} est construite à partir de la fonction p^i en ajoutant à cette dernière un hyperplan d'appui supplémentaire. Cela peut s'exprimer formellement de la façon suivante:

$$p^{i+1}(y) = \max_{1 \le j \le i+1} \{g_j(y)\}$$

$$= \max \{p^i(y), g_{i+1}(y)\}$$

où $g_j(y)$ est l'hyperplan d'appui généré à la $j^{ème}$ itération au $j^{ème}$ point d'itération y^j

c'est-à-dire:

$$g_j(y) = f^r(y^j) + s_j^T (y - y^j) \qquad \text{avec } s_j \in \delta f^r(y^j)$$

La suite de ce paragraphe est consacrée à l'établissement de propriétés qui aboutiront à la preuve de la convergence et de la terminaison de l'algorithme.

Théorème 1.5.1. :

Soit p^i la fonction linéaire par morceaux construite à la $i^{ème}$ itération de l'algorithme 1.4.2.,

Alors,

$$p^i(y) \le f^r(y) \quad \forall\, y \in Y$$

preuve :

Comme on l'a vu au début de ce paragraphe,

$$p^i(y) = \max_{1 \le j \le i} \{g_j(y)\}$$

où g_j est un hyperplan d'appui de la fonction objectif f^r, généré à la $j^{ème}$ itération de l'algorithme.

Or, par définition d'un hyperplan d'appui (déf. 1.2.1.3.),

$$g_j(y) \le f^r(y) \qquad \forall\, y \in Y$$

Vu que cette propriété est vraie $\forall$ j t.q. $0 \le j \le i$, on a dès lors que

$$\max_{0 \le j \le i} \{g_j(y)\} \le f^r(y) \qquad \forall\, y \in Y$$

c-à-d

$$p^i(y) \le f^r(y) \qquad \forall\, y \in Y$$

■

Commentaire :

Ce théorème nous montre que le modèle linéaire par morceaux de la fonction objectif reste toujours sous le graphe de f^r sur l'ensemble discret Y.

Justifions à présent le second critère d'arrêt de l'algorithme 1.4.2.

Théorème 1.5.2. :

Soit $T = \{y^j \in Y,\ j = 1..i\}$, l'ensemble des points itératifs générés par l'algorithme 1.4.2. avant la $i^{ème}$ itération.

Si

$$\exists\ j < i \ \text{ t.q. } \ y^i = y^j\ ,$$

Alors

y^i est solution du problème (P1).

preuve :

Soit j, l'entier t.q. $j < i$ et $y^j = y^i$

Par sa détermination dans l'algorithme 1.4.2.,

$$y^i = \text{argmin}\ \{\ p^{i-1}(y) \mid y \in Y\ \}\ .$$

Donc,

$$p^{i-1}(y^i) \leq p^{i-1}(y) \qquad \forall\ y \in Y.$$

D'autre part, par le théorème 1.5.1.,

$$p^{i-1}(y) \leq f^r(y) \qquad \forall\ y \in Y.$$

On en conclut

$$p^{i-1}(y^i) \leq f^r(y) \qquad \forall\ y \in Y \qquad (1)$$

En particulier,

$$p^{i-1}(y^i) \leq f^r(y^i) \qquad (2)$$

Mais, par définition,

$$p^{i-1}(y^i) = \max_{1 \leq k \leq i-1} \{g_k(y^i)\}$$

$$\geq g_j(y^i) \qquad (a)$$

$$= g_j(y^j) \qquad (b)$$

$$= f^r(y^j) \qquad (c)$$

$$= f^r(y^i) \qquad (b)$$

(a): car $j < i$

(b): car $y^i = y^j$

(c): car l'hyperplan d'appui g_j en y^j vaut la valeur de f^r en ce point, par la définition de l'hyperplan.

Donc,

$$p^{i-1}(y^i) \geq f^r(y^i) \qquad (3)$$

Les points (2) et (3) impliquent que

$$p^{i-1}(y^i) = f^r(y^i)$$

A présent, en remplaçant $p^{i-1}(y^i)$ dans la relation (1), on obtient:

$$f^r(y^i) \leq f^r(y) \qquad \forall\ y \in Y$$

■

Commentaire :

Ce résultat nous fournit notre deuxième critère d'arrêt dans l'algorithme 1.4.2.:

si on retombe sur un point déjà rencontré, c'est qu'on a atteint l'optimum.

Montrons, pour finir, que l'algorithme 1.4.2. se termine en un nombre fini d'opérations.

Théorème 1.5.3. :

L'algorithme 1.4.2. se termine en un nombre fini d'opérations.

preuve :

Cela découle immédiatement du théorème 1.5.2. et du fait qu'il n'y a qu'un nombre fini de points distincts dans Y.

■

Nous avons à présent obtenu un algorithme conceptuel qui converge et qui se termine. Il reste à voir:

★ comment résoudre le sous-problème de minimax discret permettant de trouver l'itérée suivante.

★ comment déterminer un " bon " hyperplan d'appui à la fonction objectif discrète f^r en ce point.

Ces aspects sont détaillés dans les deux paragraphes suivants.

1.6. Résolution du sous-problème de minimax discret :

1.6.1. Introduction :

Ce paragraphe a pour but de voir comment résoudre les sous-problèmes discrets de minimax de l'algorithme 1.4.2.

Le problème est formulé comme suit:

$$\min_{y \in Y} \{ p^i(y) = \max_{j=0..i} g_j(y) \}$$

ce qui est équivalent à

$$\text{(SP)} \qquad \begin{array}{lll} \min & \eta & \\ s.c. & \eta \geq g_j(y) & j=0..i \\ & y \in Y & \end{array}$$

où
- g_j est la $j^{ème}$ fonction d'appui générée par l'algorithme
- l'indice i indique que le sous-problème est celui obtenu à la $i^{ème}$ itération.

Le problème (SP) est un problème de programmation linéaire discrète avec une seule variable continue. Il peut être résolu grâce à, par exemple, une méthode d'énumération telle que la méthode "Branch and Bound" .[1]

D'autre part, on a pu observer au paragraphe 1.5. que la fonction p^{i+1} est générée en ajoutant un hyperplan d'appui à p^i. Cela implique que le problème (SP) de la $i^{ème}$ itération et celui de la $(i+1)^{ème}$ sont quasi les mêmes, sauf que celui de la $(i+1)^{ème}$ itération a une contrainte supplémentaire.

Donc, pour résoudre le problème (SP) à la $(i+1)^{ème}$ itération, des résultats obtenus pour la $i^{ème}$ itération pourraient être utilisés afin de diminuer le nombre total des calculs.

Dans la suite de cette section, on va ébaucher une procédure "Branch and Bound" permettant de résoudre le problème (SP). Des détails supplémentaires sur cette méthode sont disponibles par exemple dans [1].

1.6.2. La procédure " Branch and Bound " :

Considérons le problème (SP):

$$(SP) \qquad \begin{aligned} \min \quad & \eta \\ s.c. \quad & \eta \geq g_j(y) \qquad j = 0..i \\ & y \in Y \end{aligned}$$

Notons y_{min} et y_{max} les valeurs minimale et maximale admissibles dans Y.

La relaxation de (SP) est alors:

$$(SPr) \qquad \begin{aligned} \min \quad & \eta \\ s.c. \quad & \eta \geq g_j(y) \qquad j = 0..i \\ & y \in [y_{min}, y_{max}] \end{aligned}$$

On voit aisément que

si z*(SP) est la valeur optimale de (SP)

si z*(SPr) est la valeur optimale de (SPr)

alors

$$z^*(SP) \geq z^*(SPr)$$

figure 1.3.

$Y = \{y^1, y^2, y^3\}$

$* = f^r(y)$

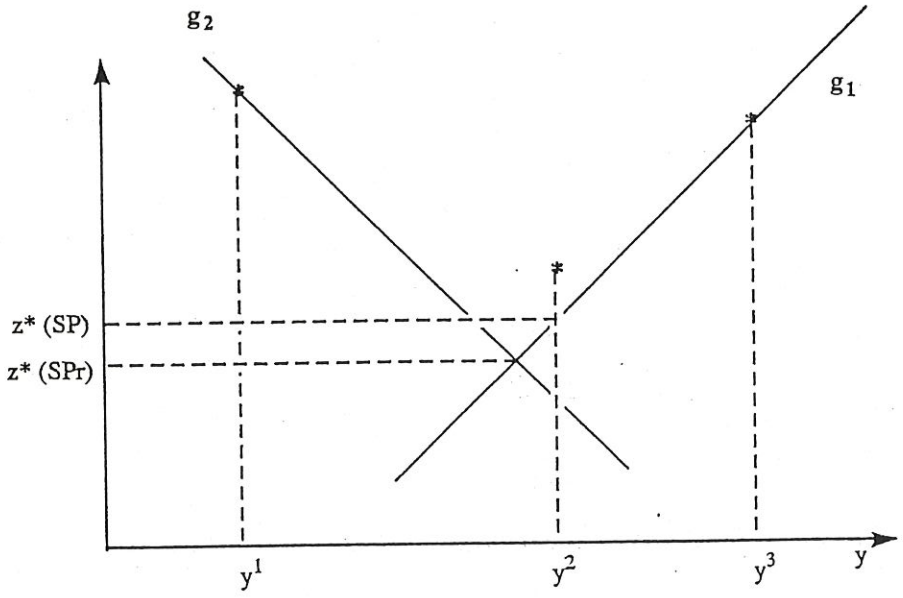


Figure 1.3. : Illustration de la propriété z* (SP) ≥ z* (SPr).

L'idée générale de la procédure " Branch and Bound " est la suivante:

- on résout la relaxation (SPr) du problème (SP), par la méthode du simplexe (cf remarque (2), p 37)

- si une solution $\bar{y}$ est obtenue, avec $\bar{y} \in Y$, l'algorithme s'arrête avec la solution optimale du problème (SP)

- sinon, une des composantes $\bar{y}_j$ (pour un certain $j \in \{1,..,m\}$) est fixée à une valeur λ discrète admissible (cf remarque (1), p 37) et deux sous-problèmes sont générés:

(SP1)

$$\begin{aligned} \min \quad & \eta \\ s.c. \quad & \eta \geq g_j(y) \qquad j=0..i \\ & y_j \leq \lambda \end{aligned}$$

et

(SP2)

$$\begin{aligned} \min \quad & \eta \\ s.c. \quad & \eta \geq g_j(y) \qquad j=0..i \\ & y_j \geq \lambda' \end{aligned}$$

où λ'= valeur admissible pour y_j directement supérieure à λ

par exemple:

$Y = D^1 \times D^2 = \{0,1,3,6\} \times \{0,2,4,7\}$

Si on fixe y_1 à 1, alors λ vaut 1

et $\lambda' = 3$.

On dit qu'on " sépare " le problème (SP).

- de façon récursive, pour chaque sous-problème, la relaxation est à nouveau résolue, et ensuite, si c'est nécessaire, on sépare en sous-problèmes.

1.6.3. Traitement des sous-problèmes :

On appelle **intermédiaire** un sous-problème pour lequel on a trouvé une solution admissible optimale qui est la meilleure solution admissible **actuellement** dans la procédure "Branch and Bound".

Il est évident que, au point de vue du temps calcul, on a intérêt à éviter (si possible) la séparation de sous-problèmes qui ne mènent pas à une meilleure solution que l'intermédiaire.

On va dès lors définir des critères de rejet d'un sous-problème candidat à la séparation.

Notons C : le candidat à la séparation

C_r : sa relaxation

z^* : la valeur optimale de l'intermédiaire

On a trois situations possibles :

1) si $z^* > z^*(C)$:
on a intérêt à garder C. Malheureusement, vu qu'on ne connaît pas $z^*(C)$, qui est la valeur optimale discrète de C, ce critère n'est pas vérifiable directement.

2) si $z^* \geq z^*(C_r)$:
on a des chances que C soit un bon candidat à la séparation, car $z^*(C) \geq z^*(C_r)$, mais ce n'est pas certain.

3) si $z^*(C_r) \geq z^*$:
on est sûr qu'on peut rejeter le problème C, car $z^*(C) \geq z^*(C_r)$

Donc, les critères de rejet sont :

1) **s'il n'existe pas de solution admissible pour (C_r)**, alors il n'y en aura sûrement pas pour C.

On rejette donc le sous-problème C.

2) si (C_r) est admissible mais que $\underline{z^*(C_r) \geq z^*}$, on rejette C.

3) si (C_r) est admissible et que **la solution optimale de (C_r) est admissible pour C** (i.e. elle appartient à Y),
alors elle est optimale pour C (i.e. $z^*(C_r) = z^*(C)$).
Dans ce cas,

si $z^*(C) \geq z^*$, on rejette C

sinon, mise à jour de l'intermédiaire.

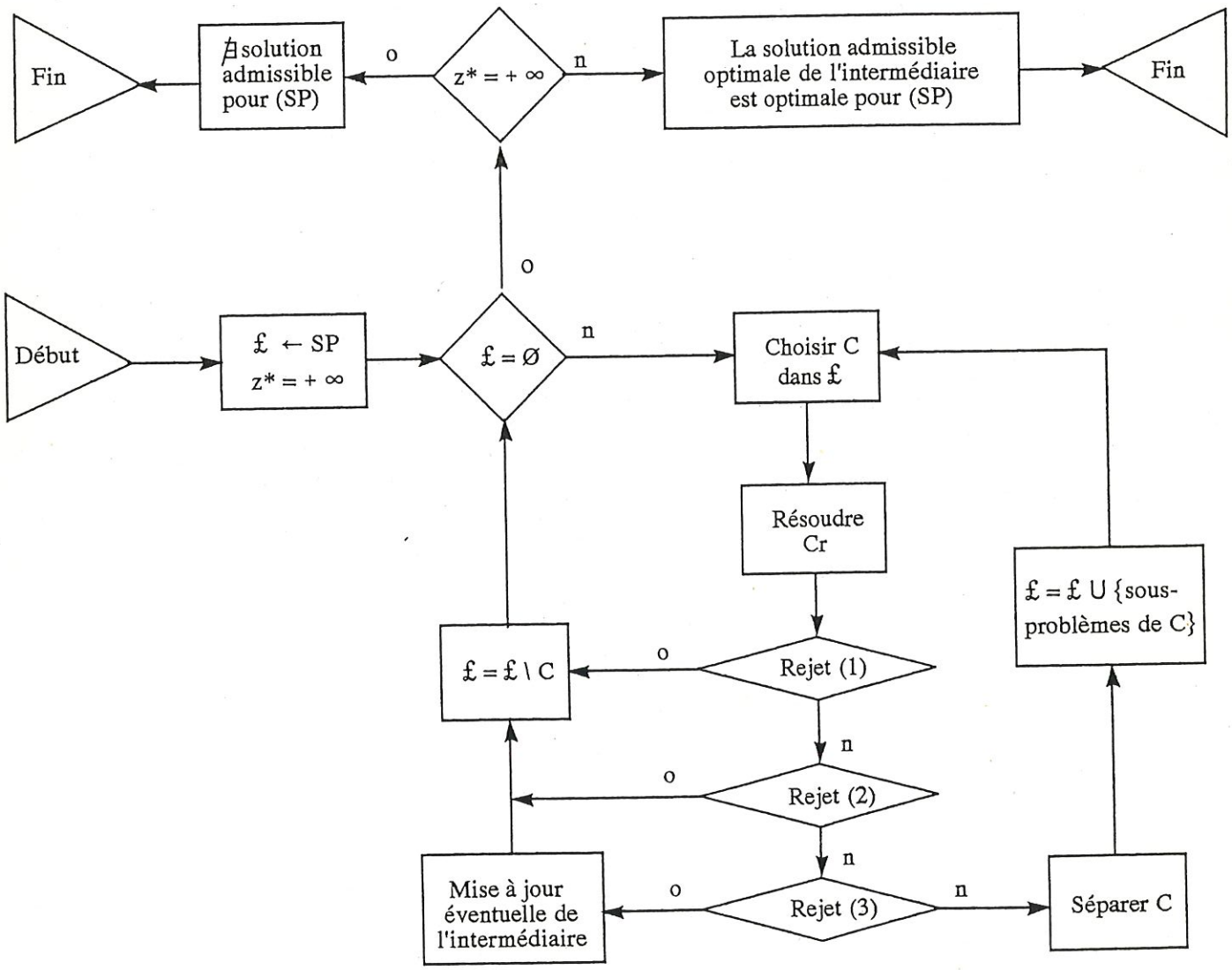
1.6.4. Schéma algorithmique général :

L'organigramme qui suit permet de visualiser la procédure " Branch and Bound ". Remarquons que l'on continue le processus de séparation décrit dans la section 1.6.3. jusqu'à ce que tous les sous-problèmes aient été éliminés ou résolus.

Posons £ = la liste des candidats à la séparation

Au départ, £ = {(SP)}

$z^* = +\infty$.



Remarques :

1) Nous n'avons pas expliqué comment choisir les variables dont la valeur va être fixée, c'est-à-dire comment choisir ce que l'on appelle la " variable de branchement " . En fait, diverses stratégies de branchement ont été développées, mais toutes procèdent d'un choix heuristique. Nous n'en dirons pas plus ici, mais des détails supplémentaires sont donnés dans [1].

2) Dans la procédure " Branch and Bound ", on a besoin de résoudre la relaxation de problèmes linéaires. On a vu d'autre part que les sous-problèmes de minimax (SP) aux étapes i et (i+1) sont pratiquement les mêmes, sauf qu'à l'étape (i+1), on a une contrainte linéaire supplémentaire. C'est d'ailleurs aussi le même type de phénomène qui se produit lorsqu'on subdivise un sous-problème (SP) via le branchement, car on ajoute une contrainte linéaire sur une variable y_j.

   La méthode duale du simplexe est bien adaptée à ce genre de problèmes, car, à partir de la base optimale du problème (SP) de la $i^{ème}$ itération, on peut "prendre un bon départ" pour le problème (SP) de la $(i+1)^{ème}$ itération. En effet, cette pratique accélère généralement les calculs (cf. [1]).

1.7. Problème du calcul " d'un meilleur " sous-gradient, pour le cas où $Y = B^m = \{0,1\}^m$:

1.7.1. Introduction :

Dans l'algorithme conceptuel, nous avons vu qu'il faut déterminer à chaque itération i, un hyperplan d'appui g_i au point courant y^i de la forme:

$$g_i(y) = f^r(y^i) + s^T (y - y^i) \qquad \text{où } s \in \delta f^r(y^i)$$

On fait cela afin de mettre à jour le modèle linéaire par morceaux de la fonction objectif.

Il faut donc déterminer un sous-gradient s de la fonction f^r au point y^i, ce qui n'est pas facile car $\delta f^r(y^i)$ est un ensemble qui n'est pas explicitement donné.

Dans le cas où la fonction f est différentiable et convexe, le théorème 1.2.2. nous fournit directement un sous-gradient: $\nabla f(y^i)$.

figure 1.4. :

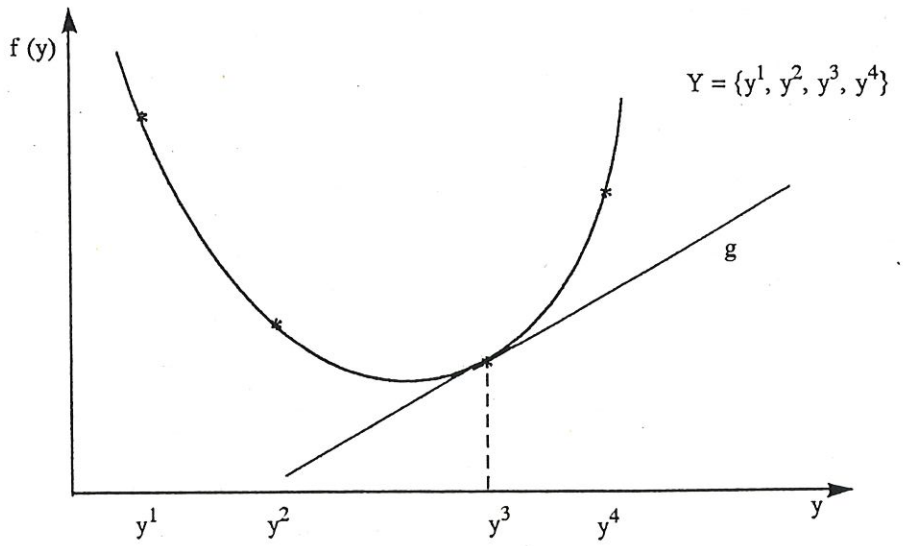


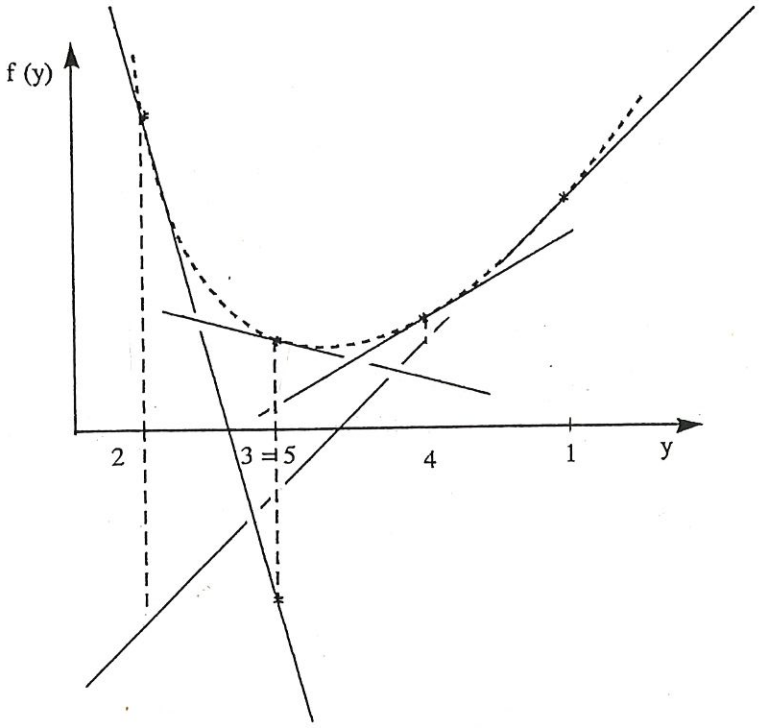
Figure 1.4. : La tangente $g(y) = f^r(y^3) + \nabla f(y^3)^T (y - y^3)$ est un hyperplan d'appui à la fonction f^r en y^3.

Cependant, il y a moyen de trouver des " meilleurs " hyperplans d'appui, c'est-à-dire des hyperplans d'appui correspondant à des sous-gradients plus proches de 0 (le "meilleur" vient du fait qu'on est à l'optimum quand $0 \in \delta f^{r}(y^{i})$, cf. théorème 1.3.1.).

" Bien " choisir les sous-gradients peut accélérer la convergence de l'algorithme. (cf. figure 1.5.)

figure 1.5. :

a)



b)

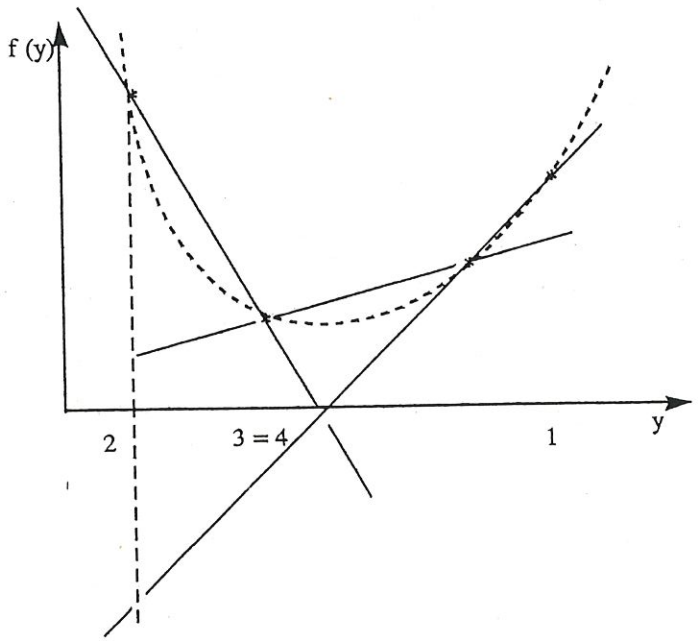


Figure 1.5. : Comparaison de la méthode du sous-gradient en utilisant un gradient (a) ou un "meilleur" sous-gradient (b); le point de départ est 1 dans les deux cas; la convergence est plus rapide en b.

On va donc essayer d'établir un processus de "relèvement" des hyperplans d'appui, basé sur la détermination particulière d'un sous-gradient, à chaque itération.

Cette méthode est établie *dans le cas particulier* d'un ensemble discret Y de la forme $Y = B^m = \{0,1\}^m$.

1.7.2. Présentation non-formelle de la méthode de "relèvement":

Le but de ce paragraphe est de trouver un moyen de déterminer à l'itération i, un sous-gradient $s_i \in \delta f^r(y^i)$ t.q. l'hyperplan d'appui correspondant, g_i, soit aussi " plat " que possible (vu que l'on est à l'optimum si $0 \in \delta f^r(y^i)$) .

Pour la simplicité des écritures, nous noterons $\bar{y}$, s et g respectivement à la place de y^i, s_i et g_i.

Afin de voir comment trouver un " bon " sous-gradient, supposons qu'on connaisse un sous-gradient quelconque s_0 où $s_0 \in \delta f^r(\bar{y})$.

On voudrait améliorer un hyperplan donné, en trouvant à partir de s_0 une nouvelle direction s

- qui resterait un sous-gradient, c-à-d telle que

  $g(y) \leq f^r(y) \quad \forall\, y \in B^m$

- qui " relève " l'hyperplan d'appui t.q. il soit le plus plat possible.

Malheureusement, on ne sait évaluer la première condition directement, car elle implique 2^m évaluations de fonction, ce qui est trop coûteux.

On va donc devoir examiner une autre méthode:

Définition 1.7.2.1. :

Considérons un point $\bar{y} \in B^m$, et un vecteur $s \in \mathbb{R}^m$.

L'ensemble de projection de s par rapport à la fonction f en $\bar{y}$, noté S, est l'ensemble suivant:

$$S = \{\, y \in \mathbb{R}^m \mid f(y) \leq g(y) \,\} \qquad \text{où } g(y) = f(\bar{y}) + s^T (y - \bar{y})$$

On peut à présent formuler la propriété suivante:

Théorème 1.7.2.2. :

Soit un vecteur $s \in \mathbb{R}^m$

s est un sous-gradient de f^r en $\bar{y}$

ssi

l'intérieur de son ensemble de projection S ne contient pas de points de B^m.

(i.e. $\nexists\, y \in B^m$ t.q. $y \in S^\circ$, où S° désigne l'intérieur de S)

figure 1.7. :

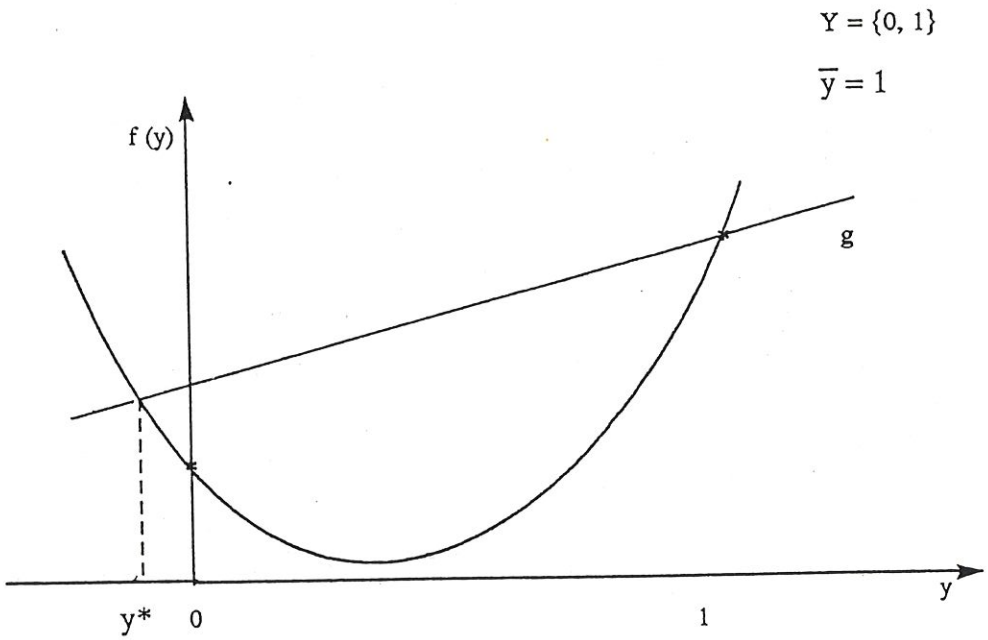


Figure 1.7. : Illustration de la propriété 1.7.2.2; S = [y*, 1] donc S^0 =]y*, 1[.
s n'est pas un sous-gradient car $0 \notin S^0$.

preuve :

$\forall\ s \in \mathbb{R}^m$, on a

s est un sous-gradient de f^r en $\bar{y}$

$\Leftrightarrow \quad g(y) \leq f^r(y) \qquad \forall\ y \in B^m$

où $g(y) = f^r(\bar{y}) + s^T (y - \bar{y})$

$\Leftrightarrow \quad \nexists\ y \in B^m$ t.q. $g(y) > f^r(y)$

$\Leftrightarrow \quad \nexists\ y \in B^m$ t.q. $y \in S°$

■

Conséquences :

La procédure de " relèvement " des hyperplans d'appui va seulement devoir veiller à ce que l'ensemble de projection S correspondant au point $\bar{y}$ courant et à la direction s choisie ne contienne pas de points de $\{0,1\}^m$.

Malheureusement, nous ne disposons pas d'un algorithme permettant de vérifier si S a bien les qualités voulues, et peut-être que, même si cet algorithme existait, il serait assez coûteux.

La version de l'algorithme du sous-gradient que nous avons étudiée va se contenter de trouver " un meilleur " hyperplan d'appui, de telle sorte que la vérification du fait que S° ne contient aucun point de B^m soit plus facilement réalisable et pas trop coûteuse.

Elle doit, pour cela, faire les hypothèses suivantes:

- **la fonction objectif f (non-restreinte à B^m) est continue, différentiable, et strictement convexe.**

Supposons, dans la suite, que le point $\bar{y}$ considéré est t.q. $\bar{y}_i = 1 \quad \forall i = 1..m$. Le raisonnement dans le cas général sera fait dans le paragraphe 1.7.3.

Analysons la situation de la figure 1.8. suivante:

figure 1.8. :

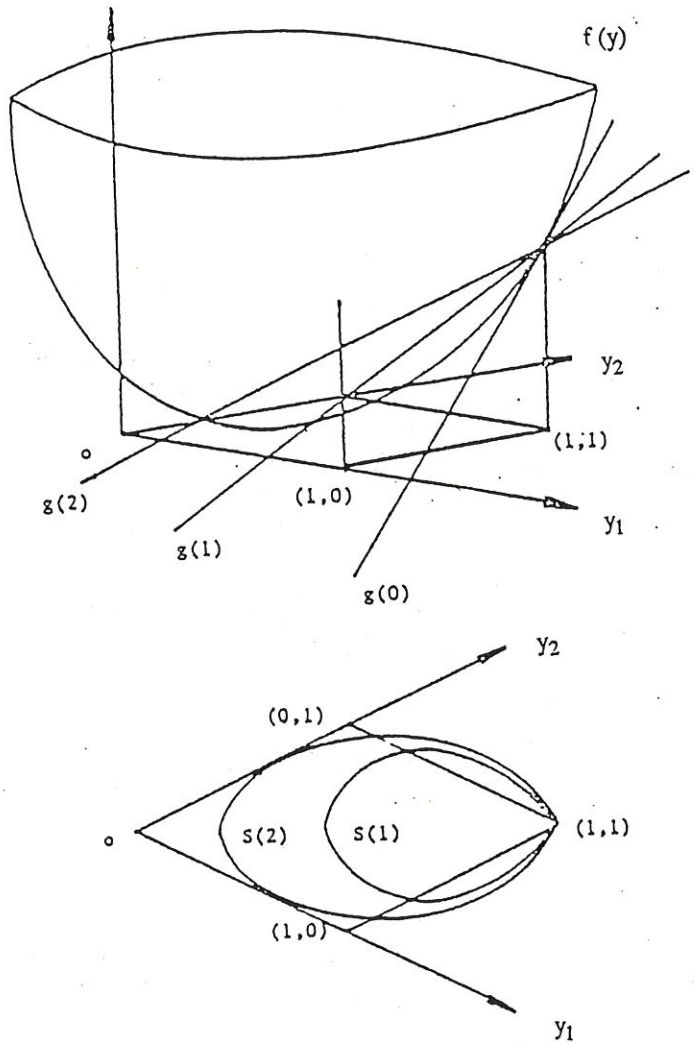


Figure 1.8.: **Processus de " relèvement " pour le calcul des sous-gradients**

Notons $A = \{y \in \mathbb{R}^m \mid y_i \geq 0 \ \ \forall i \}$

Pour $s_0 = \nabla f(\bar{y})$, on obtient l'hyperplan d'appui g_0.

Si on met à jour s_0 pour relever g_0 un petit peu, on obtient g_1 et son ensemble de projection S_1.

On voit que s_1 reste un sous-gradient car S_1° ne contient pas de points de $\{0,1\}^m$. Cette propriété reste vraie tant que S est inclu dans A.

On observe donc que :

pour tout ensemble de projection S obtenu par le processus de relèvement, l'intérieur de S, noté S°, ne contient pas de point de $\{0,1\}^m$ tant que $S \subseteq A$.

Dans ce cas-ci, on obtiendra un " bon " sous-gradient si on fait varier s jusqu'à obtenir un hyperplan d'appui tel que S touche la frontière de A.

Remarquons qu'on pourrait obtenir des hyperplans d'appui " meilleurs " que l'hyperplan d'appui finalement obtenu. Néanmoins, à chaque étape du relèvement, pour vérifier qu'on a toujours un sous-gradient, il suffit que S soit inclu dans A, ce qui peut être obtenu relativement facilement.

En effet, considérons s, un sous-gradient mis à jour quelconque, et S, son ensemble de projection.

On va établir une mesure d_i de la distance entre S et la frontière de A , dans chaque direction des axes de coordonnées, c'est-à-dire dans les directions y_i, i=1..m. (Cela sera fait en détail dans la section 1.7.4.)

Dès lors, on pourra formuler mathématiquement le processus pour calculer les sous-gradients comme le problème d'optimisation suivant:

$$\min_{s \in \mathbb{R}^m} \quad \| d(s) \|$$

$$s.c. \quad d_i(s) \geq 0 \quad \forall\, i \in \{1..m\}$$

$$où\ d(s) = (d_1, d_2, \ldots, d_m)$$

Ce problème peut être abordé avec des méthodes générales d'optimisation continue non-linéaire (cf section 1.7.5.).

Reste encore le problème du calcul de d(s) pour un vecteur s quelconque. Ce calcul sera détaillé dans le paragraphe 1.7.4 . On va y montrer qu'on peut calculer des points extrêmes de S le long de chaque direction y_i. Dans ce cas, la mesure de $d_i(s)$ sera obtenue en évaluant la distance entre le point extrême de S le long de la direction y_i et la frontière de A.

1.7.3. Une autre condition nécessaire et suffisante d'optimalité :

Dans le cas particulier où l'ensemble discret admissible Y vaut $B^m=\{0,1\}^m$, on peut énoncer une autre condition nécessaire et suffisante d'optimalité, plus facile à vérifier que de voir si $0 \in \delta f^r(\bar{y})$.

Elle s'énonce de la façon suivante:

Théorème 1.7.3. :

Soit y*, un point de l'ensemble discret admissible B^m

$$y^* \text{ minimise } f^r \text{ sur } B^m$$

$$\underline{\text{ssi}}$$

$$\exists\, t \in \delta f^r(y^*) \text{ t.q.} \qquad \begin{array}{l} t_i \leq 0 \;\; \forall\, i \text{ t.q. } y_i^* = 1 \\ t_i \geq 0 \;\; \forall\, i \text{ t.q. } y_i^* = 0 \end{array}$$

preuve :

$\boxed{\Rightarrow}$ Cela découle immédiatement du théorème 1.3.1.

En effet,

$$y^* \text{ minimise } f^r \text{ sur } B^m \Rightarrow 0 \in \delta f^r(y^*)$$

Il suffit donc de choisir $t = 0$,

ce qui vérifie bien $\quad t_i \leq 0 \quad \forall\, i \text{ t.q. } y_i^* = 1$

$\qquad\qquad\qquad\qquad\quad t_i \geq 0 \quad \forall\, i \text{ t.q. } y_i^* = 0$

$\boxed{\Leftarrow}$ On va encore se servir du même théorème.

Par hypothèse,

$\exists\, t \in \delta f^r(y^*)$ t.q. $\quad t_i \leq 0 \quad \forall\, i \text{ t.q. } y_i^* = 1$

$\qquad\qquad\qquad\qquad\qquad\quad t_i \geq 0 \quad \forall\, i \text{ t.q. } y_i^* = 0$

Dès lors,

$$t^T (y - y^*) \leq f^r(y) - f^r(y^*) \qquad \forall\, y \in B^m \qquad (1)$$

(car $t \in \delta f^r(y^*)$)

et on vérifie aisément que

$$0 \leq t^T (y - y^*) \qquad \forall\, y \in B^m \qquad (2)$$

En effet,

si $y_i^* = 1$, alors $t_i \leq 0$,

et $\quad y_i - y_i^* = 0$ ou -1.

Donc, $t_i (y_i - y_i^*) \geq 0$.

si $y_i^* = 0$, alors $t_i \geq 0$,

et $\quad y_i - y_i^* = 0$ ou 1.

Donc, $t_i (y_i - y_i^*) \geq 0$.

On déduit de (1) et de (2) que

$$0^T (y - y^*) \leq t^T (y - y^*) \leq f^r(y) - f^r(y^*) \qquad \forall\, y \in B^m$$

c'est-à-dire

$0 \in \delta f^{r}(y^*)$

ce qui, par le théorème 1.3.1, implique que
y^* minimise f^{r} sur B^m.

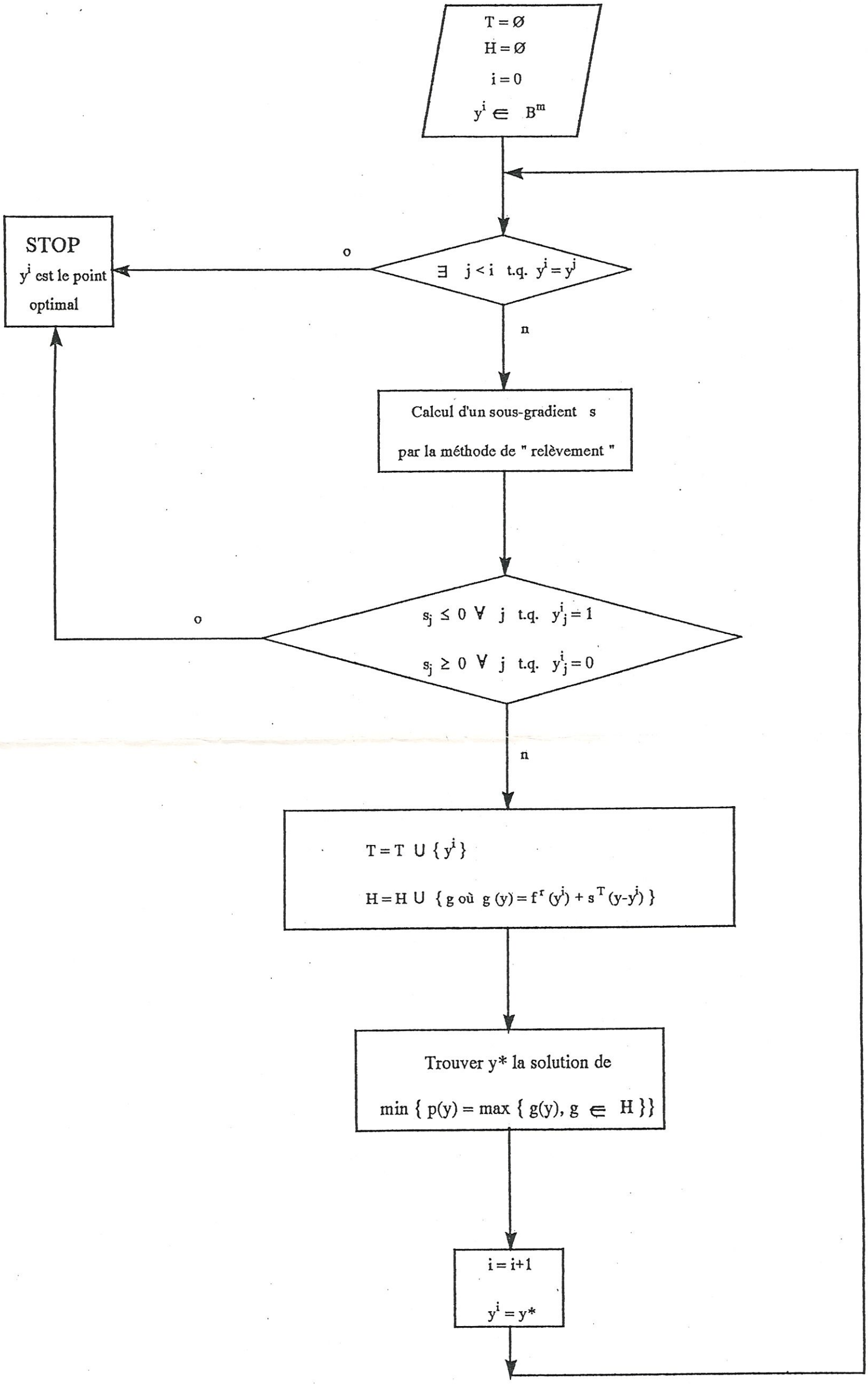
■

Remarquons que le processus pour calculer les sous-gradients intervient aussi pour le test d'optimalité. En effet, à chaque étape de relèvement, quand on obtient un sous-gradient s, l'algorithme teste pour $\bar{y}$ la condition nécessaire et suffisante du théorème 1.7.3.

Si la condition est satisfaite, l'algorithme s'arrête et le point courant $\bar{y}$ est la solution optimale.

Nous pouvons à présent schématiser l'algorithme du sous-gradient par l'organigramme qui suit.

Notons T , l'ensemble des points de B^m déjà considérés
H , l'ensemble des hyperplans d'appui déjà générés


T = Ø
H = Ø
i = 0
$y^i \in B^m$
STOP
y^j est le point optimal
o
$\exists$ j < i t.q. $y^i = y^j$
n
Calcul d'un sous-gradient s
par la méthode de " relèvement "
o
$s_j \leq 0 \; \forall$ j t.q. $y^i_j = 1$
$s_j \geq 0 \; \forall$ j t.q. $y^i_j = 0$
n
T = T U { y^i }
H = H U { g où g (y) = $f^r(y^j) + s^T(y-y^j)$ }
Trouver y* la solution de
min { p(y) = max { g(y), g $\in$ H }}
i = i+1
$y^i = y^*$

1.7.4. Traitement formel :

Hypothèses de départ :

- l'ensemble discret $Y = B^m = \{0,1\}^m$
- la fonction objectif f est continue, différentiable et strictement convexe.

Définition 1.7.4.1. :

Soit $\bar{y}$,un point dans B^m.

Soit g: un hyperplan d'appui à f^r en $\bar{y}$

h: un hyperplan obtenu en mettant à jour le sous-gradient définissant g

On dit que *h est " relevé " à partir de g*

si

1) $g(y) \leq h(y) \quad \forall\, y \in B^m$
2) $\exists\, y \in B^m$ tel que $g(y) < h(y)$

Définition 1.7.4.2. :

Soit s un vecteur de $\mathbb{R}^m$ et $\bar{y}$,un point dans B^m.

L'*ensemble de projection de s* par rapport à la fonction f en $\bar{y}$ est l'ensemble noté S, défini de la façon suivante:

$$S = \{ y \in \mathbb{R}^m \mid f(y) \leq g(y) \}$$

où $g(y) = f^r(\bar{y}) + s^T (y - \bar{y})$

Voyons une propriété de cet ensemble de projection:

Théorème 1.7.4.3. :

Soit s un vecteur de $\mathbb{R}^m$

Si la fonction f est convexe

Alors S est convexe.

preuve :

Considérons deux points de S, soient y et y'.

Montrons que :

$\forall \lambda \in [0,1] \qquad \lambda y + (1 - \lambda) y' \in S$

c-à-d

$\forall \lambda \in [0,1] \qquad f(\lambda y + (1 - \lambda) y') \leq g(\lambda y + (1 - \lambda) y')$

Or,

$$f(\lambda y + (1 - \lambda) y') \leq \lambda f(y) + (1 - \lambda) f(y') \qquad (a)$$
$$\leq \lambda g(y) + (1 - \lambda) g(y') \qquad (b)$$
$$= g(\lambda y + (1 - \lambda) y') \qquad (c)$$

car (a): f est convexe

(b): y et y' $\in$ S

(c): g est linéaire

■

L'ensemble de projection S nous permet d'établir une condition nécessaire et suffisante pour que le vecteur s auquel il correspond soit un sous-gradient de la fonction objectif f^r au point courant $\bar{y}$:

Théorème 1.7.4.4. :

Soit $s \in \mathbb{R}^m$ et $\bar{y} \in B^m$

On a:

$s \in \delta f^r(\bar{y})$ ssi $y \notin S^\circ \quad \forall\, y \in B^m$

preuve :

⇐

Par hypothèse, $\forall\, y \in B^m : \; y \notin S^\circ$

c-à-d $\forall\, y \in B^m : \; f(y) \geq g(y)$

où $g(y) = f^r(\bar{y}) + s^T (y - \bar{y})$

Or, sur B^m, $f = f^r$, donc

$\forall\, y \in B^m : \; f^r(y) \geq f^r(\bar{y}) + s^T (y - \bar{y})$

Et donc, par définition du sous-gradient,

$s \in \delta f^r(\bar{y})$.

⇒

Par hypothèse, $s \in \delta f^r(\bar{y})$

Donc, $f^r(y) \geq f^r(\bar{y}) + s^T (y - \bar{y})$ $\forall\, y \in B^m$

i.e. $g(y) \leq f^r(y)$ $\forall\, y \in B^m$

i.e. $g(y) \leq f(y)$ $\forall\, y \in B^m$

Donc,

$y \notin S^\circ \quad \forall\, y \in B^m$

■

Cette condition n'est cependant pas facile à vérifier. C'est pourquoi nous allons utiliser une autre condition, qui n'est que suffisante, mais qui est facile à réaliser.

Cette condition suffisante est la suivante:

Théorème 1.7.4.5. :

Soit $\bar{y}$, un vecteur de B^m

Soit $\bar{c}$, un vecteur de B^m, t.q. $\bar{c}_i = 1-\bar{y}_i \quad i=1..m$

Soit $A = \{y \in \mathbb{R}^m \mid y_i \leq \bar{c}_i \quad \forall\, i \text{ t.q. } \bar{c}_i = 1$

$y_i \geq \bar{c}_i \quad \forall\, i \text{ t.q. } \bar{c}_i = 0\}$

Alors, $\forall\, s \in \mathbb{R}^m$,

$s \in \delta f^r(\bar{y})$ si $S \subseteq A$

(où S = ensemble de projection de s par rapport à f en $\bar{y}$)

preuve :

On voit facilement que $\bar{y}$ est le seul point de B^m contenu dans l'intérieur de A (c'est à dire dans A°).

En effet, $\forall\, y \in B^m$,

$y \in A^\circ \Leftrightarrow \quad y_i < 1 \quad \forall\, i \text{ t.q. } \bar{y}_i = 0$

et $y_i > 0 \quad \forall\, i \text{ t.q. } \bar{y}_i = 1$

or, $y_i = 0$ ou $y_i = 1 \quad \forall\, i$

donc, $y_i = \bar{y}_i \quad \forall\, i$

Donc, $y \in A^\circ \Leftrightarrow y = \bar{y}$

Comme $S \subseteq A$ par hypothèse, on a que $S^\circ \subseteq A^\circ$.

Donc, $\bar{y}$ est le seul point de B^m qui puisse être contenu dans S°. (1)

Or, $\bar{y}$ est un point frontière de S.

En effet, $S = \{y \in \mathbb{R}^m \mid f(y) \leq g(y)\}$

où $g(y) = f^r(\bar{y}) + s^T (y - \bar{y})$

Donc, comme $g(\bar{y}) = f^r(\bar{y}) = f(\bar{y})$,

$\bar{y} \notin S^\circ$ (2)

De (1) et (2) on déduit que

$y \notin S^\circ \qquad \forall\, y \in B^m$,

c'est-à-dire, par le théorème 1.7.4.4, $s \in \delta f^r(\bar{y})$ ■

Pour vérifier que $S \subseteq A$, on regarde la distance entre S et la frontière de A, le long de chaque direction y_i, i=1..m.

Soit d_i (i=1..m) cette distance.

Il est évident que

$$S \subseteq A \quad \Leftrightarrow \quad d_i \geq 0 \quad \forall\, i$$

figure 1.9. :

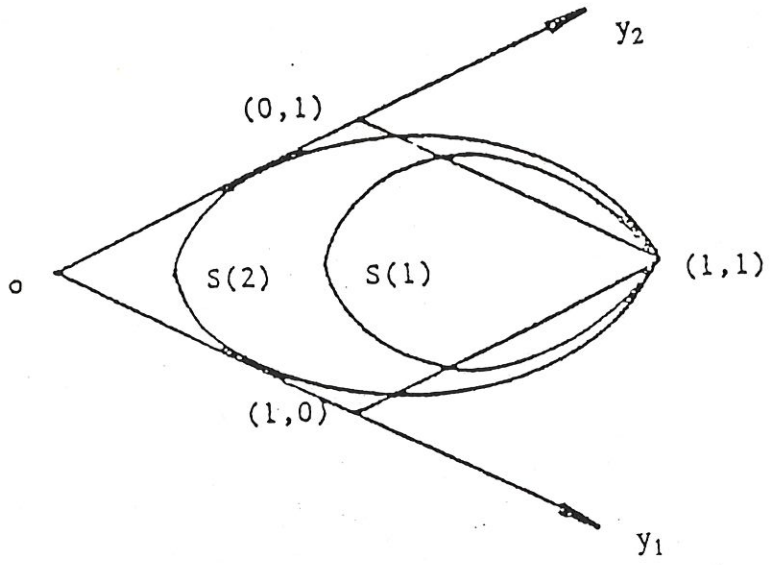


Figure 1.9.: Illustration de l'implication $S \subseteq A \Leftrightarrow d_i \geq 0 \quad \forall i$

Donc,

$$d_i \geq 0 \quad \forall i \quad \Rightarrow \quad S \subseteq A \quad \Rightarrow \quad s \in \delta f^r(\bar{y}).$$

Il suffit donc de choisir un ensemble S tel que $d_i \geq 0 \quad \forall i$ pour être sûr que le vecteur s auquel il correspond soit un sous-gradient de la fonction f^r en $\bar{y}$.

Dans cette optique, le processus de relèvement pour calculer " un meilleur " sous-gradient peut se formuler comme suit:

$$\text{(SG)} \qquad \min_{s \in \mathbb{R}^m} \ \|d(s)\|$$

$$s.c. \quad d_i(s) \geq 0 \quad \forall i \in \{1..m\}$$

$$\text{où } d(s) = (d_1, d_2, \ldots, d_m)$$

Remarque :

Le vecteur d(s) dépend du choix de s. En effet, chaque $s \in \mathbb{R}^m$ va donner un ensemble de projection S différent et donc, cela va influencer la valeur du vecteur d.

Le problème décrit ci-dessus peut être abordé via des techniques d'optimisation continue. Nous en verrons une approche possible au paragraphe 1.7.5.

Mais avant cela,

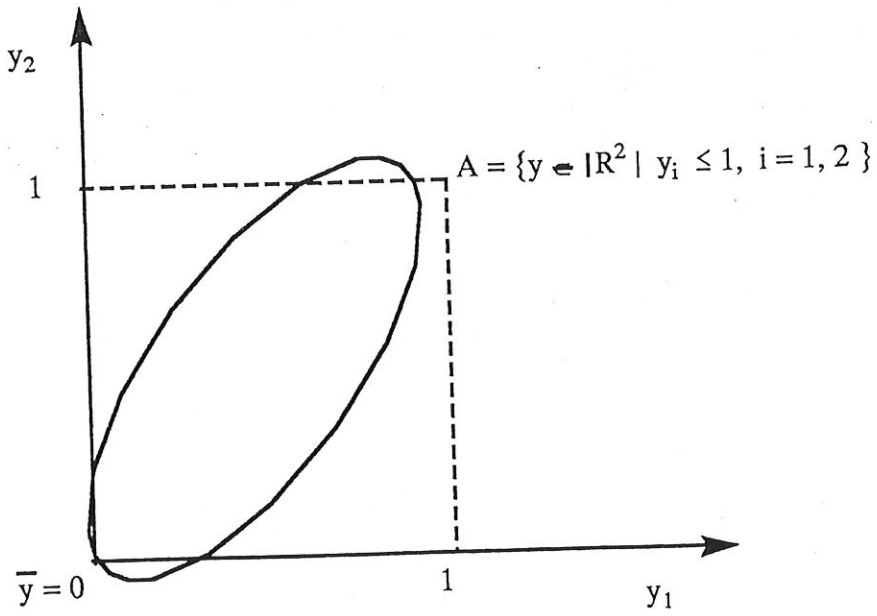
(a) voyons comment déterminer pour un sous-gradient s donné, les distances d_i, dans chaque direction y_i (i=1..m)

(b) prouvons que le problème (SG) a bien un sens, c'est-à-dire que, pour un vecteur s donné, le vecteur d(s) existe et est bien défini de façon unique.

Remarque :

Il se pourrait qu'on ait des ensembles S, dont l'intérieur ne contient pas de points de $\{0,1\}^m$, c'est-à-dire correspondant à des sous-gradients de la fonction f^r en $\bar{y}$, mais qui ne sont pas inclus dans A!

exemple:

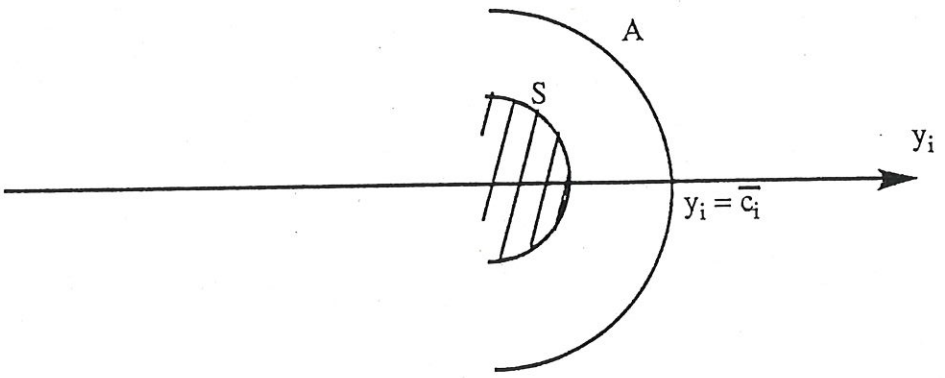


Les sous-gradients correspondant à ceux-ci sont "meilleurs" que celui qu'on trouvera en résolvant le problème (SG). Cependant, on préfère avoir un " bon " sous-gradient, qui soit relativement facile à déterminer, plutôt que de trouver " le meilleur ".

a) Détermination de la distance $d_i(s)$:

On va déterminer pour chaque i (i=1..m) la distance d_i, en recherchant un point extrême de S le long de la direction y_i et en calculant la distance entre ce point et $y_i = \bar{c}_i$, la frontière de A le long de cette direction.

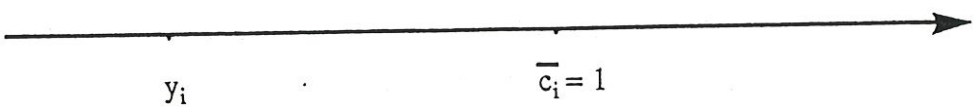
Essayons de formaliser cela, en observant le modèle ci-dessous:



Il y a deux cas possibles:

- $\bar{c}_i = 1 - \bar{y}_i = 1$ (si $\bar{y}_i = 0$)

  et $S \subseteq A$, donc tout point de S doit avoir sa $i^{ème}$ coordonnée t.q. $y_i \leq 1$



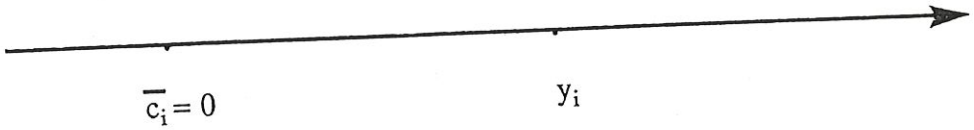
Donc, la distance entre chaque point de S et la frontière de A dans la direction y_i vaut $1 - y_i$.

Pour trouver d_i, il faut donc minimiser cette distance, c'est-à-dire:

$d_i(s) = \min (- y_i)$

- $\bar{c}_i = 1 - \bar{y}_i = 0$ (si $\bar{y}_i = 1$)

et $S \subseteq A$, donc tout point de S doit avoir sa $i^{ème}$ coordonnée t.q. $y_i \geq 0$



Donc, la distance entre chaque point de S et la frontière de A dans la direction y_i vaut y_i .

Pour trouver d_i, il faut donc minimiser cette distance, c'est-à-dire:

$d_i(s) = \min\ y_i$

Donc, si on minimise $y_i - 2\ \bar{c}_i\ y_i$, on envisage bien les deux cas possibles. En effet,

$y_i - 2\ \bar{c}_i\ y_i = y_i$ si $\bar{c}_i = 0$

$= -y_i$ si $\bar{c}_i = 1$.

Le calcul de $d_i(s)$ sera donc fait en résolvant le problème:

$$\min \quad y_i - 2\,\overline{c}_i y_i$$
$$s.c. \quad y \in S$$

ou encore:

$$\min \quad y_i - 2\,\overline{c}_i y_i$$
$$s.c. \quad f(y) - g(y) \leq 0$$

(Di)

avec $g(y) = f^{r}(\bar{y}) + s^{T}\ (y - \bar{y})$

La solution optimale de ce problème nous donne un point extrême de S le long de la direction y_i, et la valeur optimale vaut $d_i(s)$.

Il s'agit donc de minimiser une fonction linéaire, avec une seule contrainte linéaire. Une méthode de résolution de ce problème est brièvement décrite dans le paragraphe 1.7.5.

Mais nous pouvons d'ores et déjà prouver que, pour un vecteur s donné, la solution de ce problème existe et est unique, ce qui a pour conséquence que le vecteur d(s) du problème (SG) existe et est bien défini de façon unique, pour n'importe quel vecteur s.

Pour la simplicité, supposons par la suite que le point courant $\bar{y}_i = 1 \quad \forall\, i \in \{1,..,m\}$.

Le problème (Di) se réduit alors à

(Di')
$$\begin{aligned} \min \quad & y_i \\ s.c. \quad & f(y) - g(y) \leq 0 \end{aligned}$$

Remarques :

- on a toujours les hypothèses faites au début, c-à-d f continue, différentiable et strictement convexe.

- on peut se restreindre à (Di'), sans perte de généralité, car le seul autre cas possible est que la fonction objectif de (Di) soit $(-y_i)$. Pour ce cas-là, le raisonnement est exactement le même!

On a le théorème suivant:

Théorème 1.7.4.6 :

Soit s un vecteur de $\mathbb{R}^m$,

S son ensemble de projection par rapport à f en y*

Si

S est fermé et borné

Alors

le problème (Di') possède une solution unique.

preuve:

L'existence de la solution du problème (Di') découle du fait que la fonction objectif de (Di') est continue et que l'ensemble S est fermé et borné.

Montrons l'unicité par l'absurde.

Supposons qu'on ait 2 solutions optimales du problème (Di'), et notons les $\bar{y}$ et $\tilde{y}$.

Dans ce cas,

$\forall \lambda \in]0,1[$ $r = \lambda \tilde{y} + (1 - \lambda) \bar{y}$ est aussi solution.

En effet,

- $\bar{y}_i = \tilde{y}_i$ = valeur optimale de (Di') et donc

  $r_i = \lambda \tilde{y}_i + (1 - \lambda) \bar{y}_i$

  $= \bar{y}_i$ = valeur optimale de (Di')

- $r \in S$ car S est convexe , par le théorème 1.7.4.3.

Cependant, comme la fonction f est strictement convexe, par hypothèse, et que $\bar{y}$ et $\tilde{y} \in S$, on a

$$f(r) = f(\lambda \tilde{y} + (1 - \lambda) \bar{y})$$
$$\leq \lambda f(\tilde{y}) + (1 - \lambda) f(\bar{y})$$
$$\leq \lambda g(\tilde{y}) + (1 - \lambda) g(\bar{y}) \qquad \text{(a)}$$

$= g(r)$ (b)

car (a): $\bar{y}$ et $\tilde{y} \in S$

(b): g est linéaire

Donc,

$r \in S^{\circ}$

ce qui est impossible puisque toute solution du problème (Di') est nécessairement un point extrême de S.

■

Ce théorème prouve que le problème (Di) nous livre une solution unique, pour un s quelconque dans $\mathbb{R}^m$.

1.7.5. Les sous-problèmes d'optimisation continue :

Dans cette section, on discute certains détails algorithmiques pour résoudre deux classes de sous-problèmes introduits dans le chapitre précédent pour calculer des sous-gradients.

Ils sont formulés comme les problèmes (SG) et (Di) suivants:

(SG)
$$\min_{s \in \mathbb{R}^m} \|d(s)\|$$
$$s.c. \quad d_i(s) \geq 0 \quad \forall\, i \in \{1..m\}$$
$$\text{où } d(s) = (d_1, d_2, \ldots, d_m)$$

(Di)
$$\min \quad y_i - 2\bar{c}_i y_i$$
$$s.c. \quad f(y) - g(y) \leq 0$$
$$\text{avec } g(y) = f^r(\bar{y}) + s^T (y - \bar{y})$$

Si on choisit la norme l_2 dans le problème (SG), on obtient un problème aux moindres carrés non-linéaire. Dans la section 1.7.5.1., on décrit une méthode de résolution possible pour cette classe de problèmes aux moindres carrés non-linéaires. La partie 1.7.5.2. est consacrée à la résolution du problème (Di).

1.7.5.1. Le problème (SG) :

Comme on l'a fait remarquer dans la section 1.7.4., on se contente, dans l'algorithme, de trouver un " bon " sous-gradient. Dès lors, on peut ne pas résoudre (SG) exactement.

On va donc imposer que la solution soit admissible pour (SG) et qu'elle rende la fonction objectif suffisamment petite, mais pas nécessairement optimale.

Pour cela, on va appliquer l'algorithme 1.7.5.1. suivant:

0. INITIALISATION :

pour i=1..m

initialiser s_i, $\bar{s}_i$ et $\underline{s}_i$

($\bar{s}_i$ et $\underline{s}_i$ sont des bornes supérieure et inférieure de s_i)

1. MISE A JOUR :

si $d_i(s) \geq 0$

alors

si $\|d(s)\|$ est suffisamment petit, STOP

sinon

pour i=1..m : $\underline{s}_i = s_i$

$s_i = s_i + (\underline{s}_i - s_i)/2$

sinon

$\forall$ i t.q. $d_i(s) < 0$: $\bar{s}_i = s_i$

$s_i = s_i - (s_i - \bar{s}_i)/2$

2. Goto 1

L'algorithme agit de la façon suivante:

d'abord, on initialise le vecteur s.

Ensuite, on calcule un " meilleur " vecteur s en ajustant les composantes de s de la façon suivante:

- si $d_i(s) \geq 0 \quad \forall\, i \in \{1,2,..,m\}$ et si $\|d(s)\|$ est suffisamment petite, on s'arrête et le vecteur s courant est celui qu'on choisit.
- si $d_i(s) \geq 0 \quad \forall\, i \in \{1,2,..,m\}$ mais que $\|d(s)\|$ est trop grande, cela signifie qu'on n'est pas assez près des bords de A et donc, on va relever l'hyperplan d'appui, en augmentant les composantes s_i.

- si $\exists\ i \in \{1,2,..,m\}$ t.q. $d_i(s) < 0$, cela veut dire que S dépasse de l'ensemble A, et donc on va diminuer chaque composante s_i pour lesquelles $d_i(s) < 0$.

Remarque :

Le désavantage de cet algorithme est qu'il pourrait converger lentement et que, lorsqu'on limite le temps de recherche, il ne procure qu'une relativement " bonne " solution.

1.7.5.2. Le problème (Di) :

Le deuxième type de sous-problèmes introduits lors de la recherche d'un sous-gradient est le problème (Di') suivant:

(Di')
$$\begin{aligned} &\min \quad y_i \\ &s.c. \quad f(y) - g(y) \leq 0 \end{aligned}$$

avec $g(y) = f^{r}(\bar{y}) + s^{T} (y - \bar{y})$

Le théorème de Kuhn-Tucker nous fournit une condition nécessaire et suffisante pour que $\tilde{y}$ soit une solution optimale de (Di'). En effet,

si on note $l^i(y,u^i) = y_i + u^i (f(y) - g(y))$ la fonction lagrangienne associée à (Di'), avec u^i un multiplicateur de Lagrange,

alors, $\forall\, i \in \{1,2,..,m\}$,

$\tilde{y}$ est solution optimale de (Di')

ssi

$\exists\, u^i \geq 0$ t.q.

- $\nabla_y\, l^i(\tilde{y},u^i) = 0$
- $u^i\,(f(\tilde{y}) - f(\bar{y}) - s^T\,(\tilde{y} - \bar{y})) = 0$
- $f(\tilde{y}) - f(\bar{y}) - s^T\,(\tilde{y} - \bar{y}) \leq 0$

ce qui est équivalent à:

$\exists\, u^i \geq 0$ t.q.

- $u^i\,(f_{y1}'(\tilde{y}) - s_1) = 0$
- $u^i\,(f_{y2}'(\tilde{y}) - s_2) = 0$
- $\vdots$
- $u^i\,(f_{y(i-1)}'(\tilde{y}) - s_{i-1}) = 0$
- $1 + u^i\,(f_{yi}'(\tilde{y}) - s_i) = 0$
- $\vdots$
- $u^i\,(f_{ym}'(\tilde{y}) - s_m) = 0$
- $u^i\,(f(\tilde{y}) - f(\bar{y}) - s^T\,(\tilde{y} - \bar{y})) = 0$
- $f(\tilde{y}) - f(\bar{y}) - s^T\,(\tilde{y} - \bar{y}) \leq 0$

Remarquons que la $i^{ème}$ condition implique que $u^i > 0$ (car si $u^i = 0$, on a $1 = 0$).

Ce résultat couplé à l'avant-dernière condition implique que

$$f(\tilde{y}) - f(\bar{y}) - s^T\,(\tilde{y} - \bar{y}) = 0\ .$$

De plus, comme $\nabla_y^2\, l^i(\tilde{y},u^i) = u^i\,\nabla^2 f(\tilde{y})$, si on impose que $\nabla^2 f(\tilde{y})$ soit défini positif, alors $\nabla_y^2\, l^i(\tilde{y},u^i)$ est défini positif.

Conclusion :

Si on note
- h(y) = f(y) - g(y)
- l(y,u) la fonction lagrangienne associée au problème (Di'), avec u le multiplicateur de Lagrange associé à la contrainte

on obtient que:

ỹ est solution optimale de (Di')

ssi

$$F(\tilde{y}, u) = \begin{bmatrix} \nabla_y l(\tilde{y}, u) \\ h(\tilde{y}) \end{bmatrix} = 0 \qquad (\tilde{D}i)$$

Pour résoudre (D̃i), on va utiliser une méthode sécante quasi-Newton avec mise à jour BFGS structurée, c'est-à-dire en tenant compte de la structure du problème.

Voyons cela dans les sections suivantes:

a) Mise à jour BFGS :

Par "méthode sécante" pour résoudre le système d'équations non-linéaires

$F(y) = 0$ où $F : \mathbb{R}^{m+1} \rightarrow \mathbb{R}^{m+1}$

on désigne la procédure itérative ci-dessous :

- résoudre $B.d = -F(y)$
- $y_+ = y + d$
- calculer $B_+ = BFGS(B)$

où
- B est une approximation de $\nabla F(y)$
- d est la direction quasi-Newton

o B_+ est la mise à jour BFGS de la matrice B, qui doit vérifier l'équation sécante

$B_+ d = z$ où $z = F(y_+) - F(y)$.

Pour déterminer la forme de B, examinons de plus près le problème. En principe,

$$\nabla F(y) = \begin{bmatrix} \nabla_y^2 l(y,u) & \nabla h(y) \\ \nabla h(y)^T & 0 \end{bmatrix}$$

Une partie des informations du premier ordre sur F(y) est calculable directement. Il s'agit ici de $\nabla h(y)$.

Par contre, on va devoir approcher la partie $\nabla_y^2 l(y,u)$.

Comme $\nabla_y^2 l(y,u) = u^i \nabla^2 f(y)$ est

- symétrique
- défini positif (si on impose $\nabla^2 f(y)$ déf. pos.),

on va approcher cette partie par la matrice B_l symétrique et définie positive.

Donc,

$$B = \begin{bmatrix} B_l & \nabla h(y) \\ \nabla h(y)^T & 0 \end{bmatrix}$$

et la méthode BFGS structurée utilisée pour résoudre (Di') peut être formulée comme la procédure itérative suivante:

- Poser

$$B = \begin{bmatrix} B_1 & \nabla h(y) \\ \nabla h(y)^T & 0 \end{bmatrix}$$

- Résoudre

$$Bd = -F(y)$$

- Faire

$$y_+ = y + d$$

- Calculer

$$B_{1+} = B_1 + \frac{y y^T}{d} - \frac{B_1 d d^T B_1}{d^T B_1 d}$$

- Poser

$$B_+ = \begin{bmatrix} B_{1+} & \nabla h(y_+) \\ \nabla h(y_+)^T & 0 \end{bmatrix}$$

Reste maintenant à résoudre le système B.d = -F(y), pour trouver le vecteur d.

b) Résolution du système B.d = -F(y) :

Dans la méthode sécante ci-dessus, on doit résoudre un système linéaire à chaque itération:

$$Bd = -F \tag{1}$$

où

$$B = \begin{bmatrix} B_1 & \nabla h \\ \nabla h^T & 0 \end{bmatrix}$$

avec B_1 symétrique et définie positive

On peut résoudre ce système de la façon suivante :

Soit $d = (v,\lambda)^T$

et $-F = (z,\beta)^T$ où $v,z \in \mathbb{R}^m$

$\lambda, \beta \in \mathbb{R}$

On récrit le système (1) comme:

$$B_1 v + \nabla h \lambda = z$$

$$\nabla h^T v = \beta$$

On résout le système ci-dessus pour v et λ comme suit :

$$v = B_1^{-1} (z - \nabla h \lambda)$$

$$\lambda = \frac{\nabla h^T B_1^{-1} z - \beta}{\nabla h^T B_1^{-1} \nabla h}$$

Tout cela est équivalent à utiliser les quatre opérations suivantes :

1) $B_1 a = \nabla h$

2) $B_1 b = z$

3) $v = b - a \lambda$

4) $\lambda = (\nabla h^T b - \beta) (\nabla h^T a)^{-1}$

Or, B_1 est symétrique et définie positive, ce qui implique que sa décomposition de Cholesky en facteurs triangulaires existe. Cette décomposition en facteurs triangulaires facilite fort la résolution de (1) et (2).

Conclusion :

Pour résoudre le système $B d = -F$,

on effectue les quatre opérations ci-dessus. L'utilisation de la factorisation de Cholesky permettra de résoudre plus facilement les deux premières opérations.

1.8. Implémentation parallèle de l'algorithme du sous-gradient:

Jusqu'ici, on n'a pas encore abordé le problème de l'implémentation de l'algorithme du sous-gradient.

Comme on l'a vu dans le chapitre 1.7., à chaque étape du processus de "relèvement", pour calculer un sous-gradient, on a m sous-problèmes à résoudre, à savoir

$$\forall\, i \in \{1,..,m\} \qquad \begin{aligned} \min \quad & y_i - 2\overline{c}_i y_i \\ s.c. \quad & f(y) - g(y) \leq 0 \end{aligned}$$

La parallélisation de ce travail permettrait un gain de temps, et cela semble possible vu que les m sous-problèmes sont indépendants et qu'ils ont tous quasi la même taille.

Une autre procédure qu'il serait intéressant de "paralléliser" est la procédure "Branch and Bound" où plusieurs sous-problèmes pourraient être résolus en même temps.

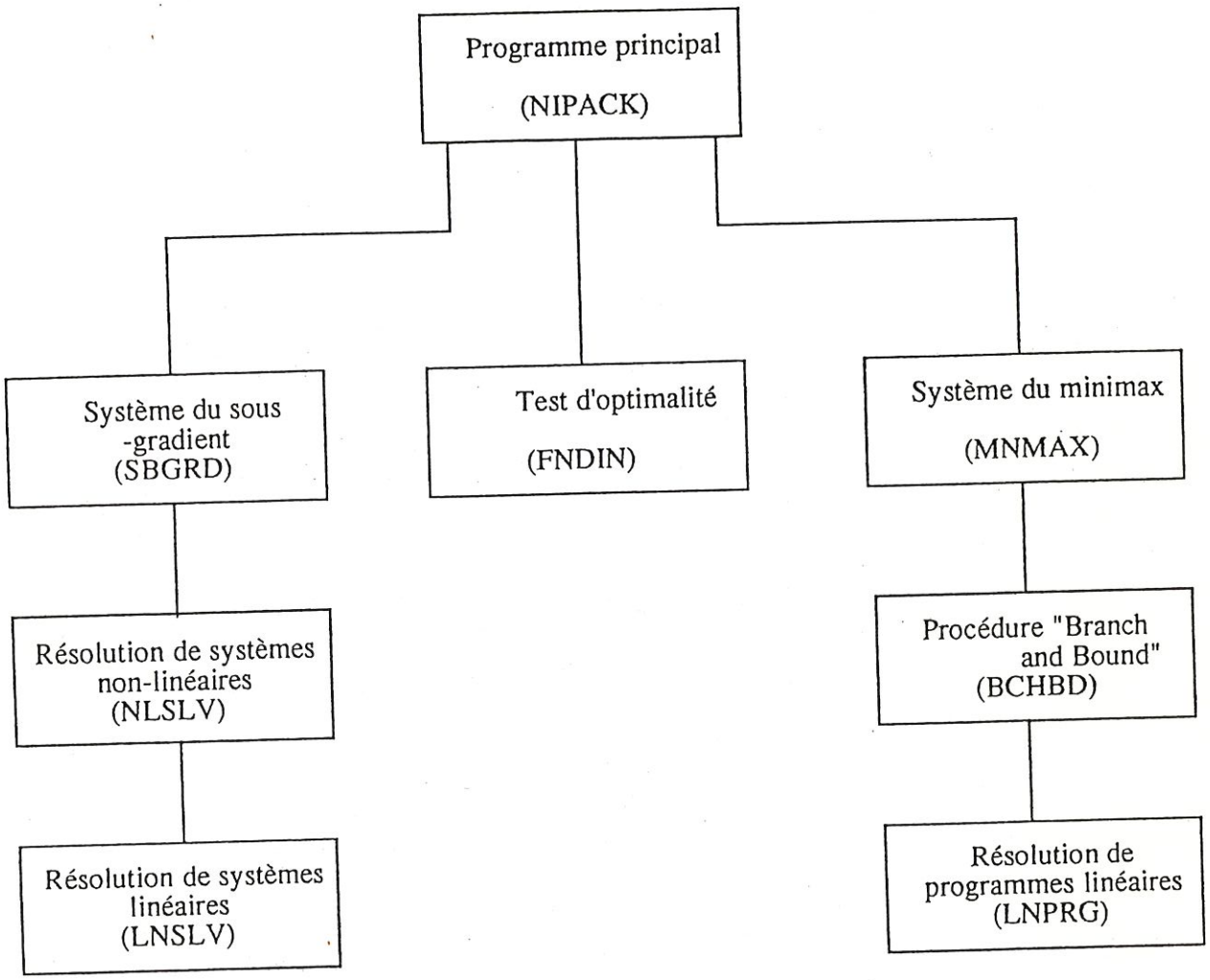
Dès lors, ZHIJUN WU (cf. [6]) a mis au point une implémentation parallèle de l'algorithme du sous-gradient, dont nous allons brièvement décrire la structure dans la suite de cette section.

Le système NIPACK :

NIPACK est un système software développé pour implémenter l'algorithme du sous-gradient. Le système est écrit en EXPRESS C et fonctionne sur des machines à mémoires parallèles.

Il contient un groupe de sous-routines parallélisées, dont les factorisations LU, QR et de Cholesky, des programmes résolvant des systèmes triangulaires, et la procédure " Branch and Bound ".

Il y a huit modules dans NIPACK, chacun contenant un groupe de sous-routines utilisées dans un but précis. Décrivons schématiquement la fonction de chacun de ces modules et les relations entre ceux-ci:


Programme principal
(NIPACK)
Système du sous -gradient
(SBGRD)
Test d'optimalité
(FNDIN)
Système du minimax
(MNMAX)
Résolution de systèmes non-linéaires
(NLSLV)
Procédure "Branch and Bound"
(BCHBD)
Résolution de systèmes linéaires
(LNSLV)
Résolution de programmes linéaires
(LNPRG)

< module 1 > NIPACK :

Programme principal pour l'algorithme du sous-gradient. Il demande à l'utilisateur un point de départ, effectue la boucle principale de l'algorithme et donne la solution.

< module 2 > FNDIN :

Test d'optimalité. Le programme teste si le point itératif courant a déjà été considéré auparavant. Si oui, retour au programme principal, avec la solution optimale.

< module 3 > SBGRD :

Calcul des sous-gradients. Le programme effectue un processus de " relèvement ", en mettant à jour les sous-gradients pour finalement en trouver un " suffisamment bon ". Il revient également au problème principal lorsqu'on trouve un sous-gradient qui vérifie la condition nécessaire et suffisante d'optimalité. Dans ce cas, une solution optimale est effectivement déterminée.

< module 4 > NLSLV :

Ce programme résout un système d'équations non-linéaires. Il est utilisé pour la détermination des points extrêmes d'un corps convexe dans les directions des axes de coordonnées, y_i (i = 1..m). Une méthode quasi-Newton avec mise à jour BFGS structurée est utilisée.

< module 5 > LNSLV :

Principales sous-routines pour le calcul matriciel. Des versions parallèles des procédures de factorisation LU, QR et de Cholesky sont implémentées, ainsi que des versions parallèles des programmes résolvant des systèmes triangulaires inférieurs ou supérieurs.

< module 6 > MNMAX :

Résout des problèmes linéaires entiers de minimax. Il appelle la procédure " Branch and Bound " pour trouver la solution optimale entière.

< module 7 > BCHBD :

Procédure récursive, résolvant des problèmes " relâchés " linéaires et créant des sous-problèmes.

< module 8 > LNPRG :

Résout des programmes linéaires continus. Il est utilisé par le module BCHBD afin de trouver la solution des relaxations linéaires créées. Cette partie est une implémentation de l'algorithme du simplexe.

1.9. **Conclusion** :

Nous avons, dans ce premier chapitre, présenté un algorithme du sous-gradient pour des problèmes de programmation discrète non-linéaire. Notre approche est basée sur l'utilisation de sous-gradients de la fonction objectif afin de linéariser celle-ci par morceaux. L'algorithme cherche la solution itérativement parmi l'ensemble des points admissibles, et à chaque itération, il génère l'itéré suivant en résolvant le problème de départ pour le modèle linéaire par morceaux. Ce dernier est construit avec des hyperplans d'appui de la fonction objectif aux points itératifs déjà générés.

Pour déterminer si le point itératif courant est optimal, on a établi deux critères d'optimalité. L'un d'eux est une condition nécessaire et suffisante d'optimalité.

Nous avons également développé une façon de déterminer les sous-gradients pour des fonctions objectifs différentiables et strictement convexes sur un ensemble discret admissible de la forme $\{0,1\}^m$. Les démonstrations formelles ont été détaillées, ainsi que certaines méthodes de résolution des sous-problèmes intervenant dans l'algorithme.

Enfin, nous avons brièvement décrit l'implémentation en parallèle de l'algorithme du sous-gradient. Celle-ci se compose d'un ensemble de sous-routines parallèles telles que les factorisations de matrices LU, QR, et de Cholesky, la résolution de systèmes triangulaires, la procédure " Branch and Bound ".

Plusieurs aspects de notre approche peuvent être améliorés. D'abord, on pourrait envisager des programmes non-linéaires en variables mixtes avec éventuellement des contraintes, linéaires ou non. Ensuite, la technique pour calculer un " bon " sous-gradient pourrait être améliorée en remplaçant la procédure de "relèvement" par une procédure qui calculerait le plus grand ensemble convexe possible correspondant à un sous-gradient et ne contenant pas de points de $\{0,1\}^m$. En effet, la procédure de "relèvement" telle qu'elle est décrite dans ce travail ne parvient pas toujours à trouver un sous-gradient qui vérifie la condition nécessaire et suffisante d'optimalité la première fois qu'on passe à l'optimum! Dans ce cas, l'algorithme ne s'arrête que la deuxième fois qu'il considère ce point optimal. Donc, si un tel programme est mis au point, il permettra d'obtenir un hyperplan d'appui " plus plat " et ainsi accélérer la convergence de l'algorithme.

Chapitre 2: Une méthode d'optimisation non-linéaire en variables mixtes, avec contraintes

2.1. Introduction :

Ce deuxième chapitre est consacré à la description d'une méthode permettant de résoudre un problème de la forme suivante:

$$(P) \qquad \begin{aligned} \min\ & f(x,y) \\ s.c.\ \ & g(x,y) \leq 0 \\ & x \in X \\ & y \in Y \end{aligned}$$

où f et g sont des fonctions de $\mathbb{R}^n \times \mathbb{R}^m$ dans $\mathbb{R}$, X est une partie de $\mathbb{R}^n$, Y est une partie discrète finie de $\mathbb{R}^m$

Dans la section 2.2., nous allons définir un problème équivalent à (P), qui sera à la base de la méthode décrite au paragraphe 2.3. Pour finir, les démonstrations de convergence de l'algorithme feront l'objet de la section 2.4.

2.2. Position du problème et problème équivalent :

Considérons à présent le problème suivant:

$$(P_0) \qquad \min_{y} \; F(y) \quad s.c. \quad y \in Y \cap V$$

où $V = \{ y \in Y \mid \exists x \in X \text{ t.q. } g(x,y) \leq 0 \}$ et F(y) est la valeur optimale du problème

$$P(y) \qquad \min_{x} \; f(x,y) \quad s.c. \quad g(x,y) \leq 0, \; x \in X$$

Le théorème 2.2.1 nous montre que les problèmes (P) et (P_0) sont équivalents :

Théorème 2.2.1 :

Soient les problèmes (P_0) et (P) décrits ci-dessus.

On a les trois équivalences suivantes:

a) (x*,y*) est solution de (P)

ssi

y* est solution de (P_0) avec x* solution optimale de P(y*)

b) (P) est non-admissible

ssi

(P_0) est non-admissible

c) (P) est non-borné

ssi

(P_0) est non-borné

preuve du point (a) :

$\boxed{\Rightarrow}$

Par hypothèse,

(x*,y*) est solution de (P)

Donc,

$g(x^*,y^*) \leq 0 \quad , \quad x^* \in X \quad \text{et} \quad y^* \in Y$

et

$\forall\ (x,y) \in X \times Y$ t.q. $g(x,y) \leq 0$, on a

$$f(x^*,y^*) \leq f(x,y) \qquad (A)$$

On doit montrer que

y* est solution de (P_0) et x* solution de P(y*),

c'est-à-dire

(1) $y^* \in Y \cap V$

(2) $\forall\ y \in Y \cap V \qquad F(y^*) \leq F(y)$

(3) $F(y^*) = f(x^*,y^*)$

On a immédiatement (1) car par hypothèse $y^* \in Y$, $x^* \in X$ et

$g(x^*,y^*) \leq 0$.

Pour prouver la relation (2), remarquons que pour tout $y \in Y \cap V$, on a :

$$F(y^*) \leq f(x^*,y^*) \leq F(y) \qquad (B)$$

Puisque $x^* \in X$ et $g(x^*,y^*) \leq 0$, la première inégalité découle de la définition de F(y*). La seconde inégalité provient de (A).

Comme $y^* \in Y \cap V$, par (2) on a $F(y^*) = \min \{ F(y) \mid y \in Y \cap V\}$ et donc les inégalités dans (B) impliquent $F(y^*) = f(x^*,y^*) = \min \{ F(y) \mid y \in Y \cap V\}$ c'est-à-dire (3).

$\boxed{\Leftarrow}$

Par hypothèse,

y^* est solution de (P_0) et x^* est solution de $P(y^*)$

Donc,

- $y^* \in Y \cap V$
- $\forall\, y \in Y \cap V \qquad F(y^*) \leq F(y)$ (C)

c'est-à-dire:

$$\begin{array}{ccc} \min f(x,y^*) & \leq & \min f(x,y) \\ \text{s.c. } g(x,y^*) \leq 0 & & \text{s.c. } g(x,y) \leq 0 \\ x \in X & & x \in X \end{array}$$

- $x^* \in X$ et $g(x^*,y^*) \leq 0$
- $\forall\, x \in X$ t.q. $g(x,y^*) \leq 0$: $f(x^*,y^*) \leq f(x,y^*)$

On doit montrer que

(1) $x^* \in X$, $y^* \in Y$ et $g(x^*,y^*) \leq 0$

et que

(2) $\forall\, (x,y) \in X \times Y$ t.q. $g(x,y) \leq 0$, on a

$f(x^*,y^*) \leq f(x,y)$

(1) est évident, vu les hypothèses.

Pour (2), considérons $(\bar{x},\bar{y}) \in X \times Y$ t.q. $g(\bar{x},\bar{y}) \leq 0$.

Par hypothèse, comme x^* est minimiseur de $P(y^*)$,

on a $F(y^*) = f(x^*,y^*)$

et donc, par (C)

$$f(x^*,y^*) = F(y^*) \leq F(\bar{y})$$

où $F(\bar{y}) = \min f(x,\bar{y})$

s.c. $x \in X$

$g(x,\bar{y}) \leq 0$

et comme $\bar{x} \in X$ et $g(\bar{x},\bar{y}) \leq 0$,

on a $F(\bar{y}) \leq f(\bar{x},\bar{y})$.

Donc,

$$f(x^*,y^*) \leq f(\bar{x},\bar{y})$$

preuve du point (b) :

$\boxed{\Rightarrow}$

Par hypothèse,

(P) est non-admissible, c'est-à-dire

$S = \{(x,y) \in X \times Y \mid g(x,y) \leq 0\} = \emptyset$

Nous devons montrer que

$\forall\, y \in Y \;\; \exists\, x \in X$ t.q. $g(x,y) \leq 0$

Soit $\bar{y} \in Y$,

et supposons par l'absurde que

$\exists\, \bar{x} \in X$ t.q. $g(\bar{x},\bar{y}) \leq 0$.

Dans ce cas,

$(\bar{x},\bar{y}) \in S$ ce qui est impossible.

$\boxed{\Leftarrow}$

Par hypothèse,

(P_0) est non-admissible, c'est-à-dire

$$S_0 = \{ y \in Y \mid \exists x \in X \text{ t.q. } g(x,y) \leq 0\} = \emptyset$$

Montrons que

$$S = \{(x,y) \in X \times Y \mid g(x,y) \leq 0\} = \emptyset$$

Supposons par l'absurde que

$$\exists (\bar{x},\bar{y}) \in X \times Y \quad \text{t.q. } g(\bar{x},\bar{y}) \leq 0.$$

Alors,

$\bar{y} \in S_0$ ce qui est impossible.

preuve de (c) :

$\Rightarrow$

Par hypothèse,

(P) est non-borné.

Donc, $\exists (x^*,y^*) \in X \times Y$ t.q. $g(x^*,y^*) \leq 0$ et

$f(x^*,y^*) = -\infty$

On a que

$y^* \in Y \cap V$ et x^* est solution de $P(y^*)$

car $f(x^*,y^*) = -\infty$

$g(x^*,y^*) \leq 0$ et $x^* \in X$

Cela implique que $F(y^*) = -\infty$

et donc que (P_0) est non-borné.

$\Leftarrow$

Par hypothèse,

(P_0) est non-borné.

Donc, $\exists y^* \in Y \cap V$ t.q. $F(y^*) = -\infty$

Donc, $\exists x^* \in X$ t.q. $g(x^*,y^*) \leq 0$ et

$f(x^*,y^*) = -\infty$

On a dès lors que

(x^*,y^*) est admissible pour (P)

et $f(x^*,y^*) = -\infty$

Cela implique que (P) est non-borné.

■

Remarque :

Notons dès maintenant que, pour assurer la convergence de la méthode, on va devoir faire les hypothèses suivantes:

- f convexe et différentiable sur $\mathbb{R}^n \times \mathbb{R}^m$
- g convexe et différentiable sur $\mathbb{R}^n \times \mathbb{R}^m$

2.3. Description de la méthode :

Les problèmes (P) et (P_0) étant équivalents, la méthode étudiée maintenant a pour objectif de résoudre (P_0) plutôt que (P). L'ensemble admissible de (P_0) étant discret et fini, la procédure consiste à minimiser F en éliminant des éléments de $Y \cap V$ jusqu'à obtenir soit un ensemble $Y^k = \emptyset$, soit la solution y* recherchée.

Pour minimiser la fonction F, on construit une linéarisation du problème de départ (P) autour du point (x^k, y^k) dont on recherche un minimum (x,y). Si $F(y) < F(y^k)$, on pose $y^{k+1} = y$ et on recommence la procédure, sinon on améliore le modèle linéaire jusqu'à l'obtention d'un point y^{k+1} qui convient. Dans le second cas, on effectue une suite de pas nuls et on engendre une suite d'itérées $y^k_1, y^k_2, \ldots, y^k_i, \ldots$. Chaque point y engendré est retiré de Y.

D'une façon plus schématique :

à l'itération k, si l'ensemble discret des points admissibles non-encore considérés Y^k est non-vide, nous avons:

- la résolution d'une approximation linéaire (M^k_i) du problème (P), afin d'obtenir un point y^k_{i+1} discret admissible
- la résolution d'un problème non-linéaire continu $P(y^k_{i+1})$. La solution de ce problème, si elle existe, est notée x^k_{i+1}
- si y^k_{i+1} est un bon candidat, c'est-à-dire si
  - y^k_{i+1} est admissible pour (P_0)

  et
  - $F(y^k_{i+1}) < F(y^k)$

  alors, $y^{k+1} = y^k_{i+1}$ est le nouvel itéré.
  On dit qu'on a effectué un **pas sérieux**.

- si y^k_{i+1} est non-admissible ou si $F(y^k_{i+1}) \geq F(y^k)$
  alors,
  on fait un **pas nul**, c'est-à-dire qu'on n'a pas encore de nouvel itéré et qu'on va devoir modifier l'approximation linéaire du problème (P) pour obtenir un autre candidat et afin que l'ancien y^k_{i+1} ne soit plus considéré.

Cette modification sera faite soit en reformulant l'ensemble des points discrets admissibles, de façon à exclure l'ancien point y^k_{i+1}, soit par une diminution du rayon de la région autour de $u^k=(x^k,y^k)$ dans laquelle on fait confiance au modèle.

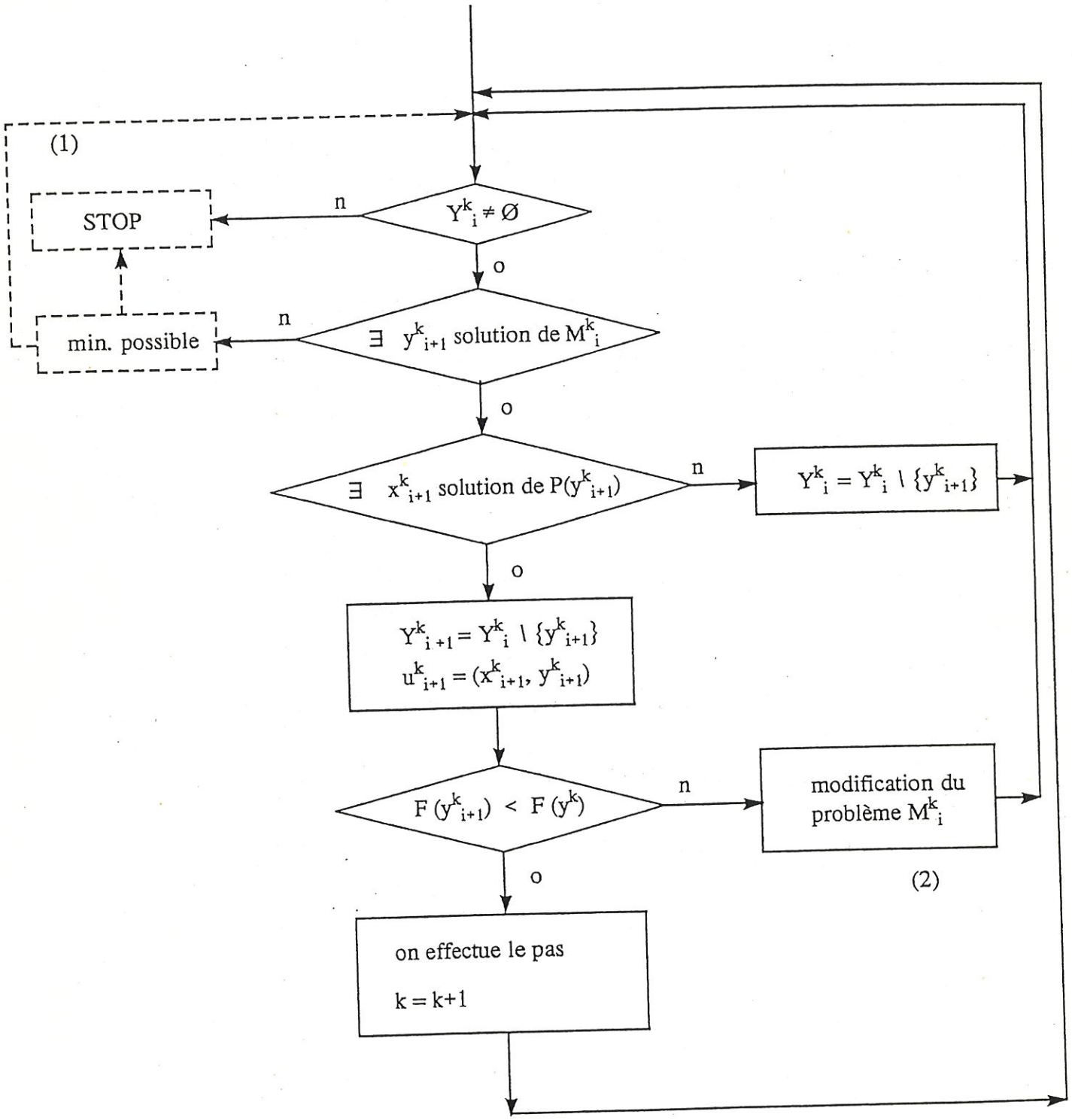
Remarque :

un point discret $\bar{y} \in Y$ est admissible pour (P_0)

ssi

le problème $P(\bar{y})$ est admissible

Visualisons le processus global sur l'organigramme suivant:


(1)
STOP
n
$Y^k_i \neq \emptyset$
o
min. possible
n
$\exists$ y^k_{i+1} solution de M^k_i
o
$\exists$ x^k_{i+1} solution de $P(y^k_{i+1})$
n
$Y^k_i = Y^k_i \setminus \{y^k_{i+1}\}$
o
$Y^k_{i+1} = Y^k_i \setminus \{y^k_{i+1}\}$
$u^k_{i+1} = (x^k_{i+1}, y^k_{i+1})$
$F(y^k_{i+1}) < F(y^k)$
n
modification du problème M^k_i
(2)
o
on effectue le pas
k = k+1

Remarques :

(1) Les deux endroits où l'algorithme peut s'arrêter avec le minimum seront expliqués dans la section suivante.

(2) Nous allons, dans la suite de ce paragraphe, détailler le problème linéaire (M^k_i) et ses modifications possibles dans le cas d'un pas nul.

Supposons qu'on se trouve au début de la $k^{ième}$ itération, avec un point $y^k = y^k_1 \in Y^k$ admissible pour (P_0), et notons x^k_1 la solution optimale du sous-problème continu $P(y^k_1)$.

Posons $u^k = u^k_1 = (x^k_1, y^k_1)$.

Nous noterons $\nabla f(u^k)$ le vecteur gradient de la fonction f en u^k.

On doit résoudre l'approximation linéaire du problème (P) suivante: (i = 1, au départ)

(M^k_i)

$$
\begin{aligned}
&\min \mu \\
&s.c. \quad f(u^k_j) + \nabla f(u^k_j)^T (u - u^k_j) \le \mu \quad j = 1..i \\
&\qquad g(u^k_j) + \nabla g(u^k_j)^T (u - u^k_j) \le 0 \quad j = 1..i \\
&\qquad \|u - u^k\| \le \rho^k_i \\
&\qquad u \in X \times Y^k_i \\
&\qquad \mu \in \mathbb{R}
\end{aligned}
$$

où $\rho^k_1 = +\infty$ et $Y^k_1 = Y^k \setminus \{y^k\}$

- Si (M^k_i) n'est pas admissible, on peut montrer qu'alors, soit u^k est un point isolé (au sens de la définition 4 de l'introduction) soit un minimiseur local ou global (au sens de la définition 5 dans l'introduction). Voir pour cela la section 2.4.

- Sinon, on obtient un candidat y^k_{i+1} potentiellement admissible, où y^k_{i+1} est la partie discrète de la solution de (M^k_i).

- Dans le cas d'un **pas sérieux**,

  i.e. si y^k_{i+1} est admissible pour (P_0)

  et $\min f(x,y^k_{i+1}) < \min f(x,y^k)$

  on pose

  $y^{k+1} = y^k_{i+1}$

  qui est donc le nouveau point courant.

- Dans le cas d'un **pas nul**,

  › si y^k_{i+1} est non-admissible pour (P_0), on reconsidère le problème (M^k_i) avec $Y^k_i = Y^k_i \setminus \{y^k_{i+1}\}$. On est donc sûr que le point y^k_{i+1} ne sera plus pris en compte.

  › si y^k_{i+1} est admissible pour (P_0) mais que

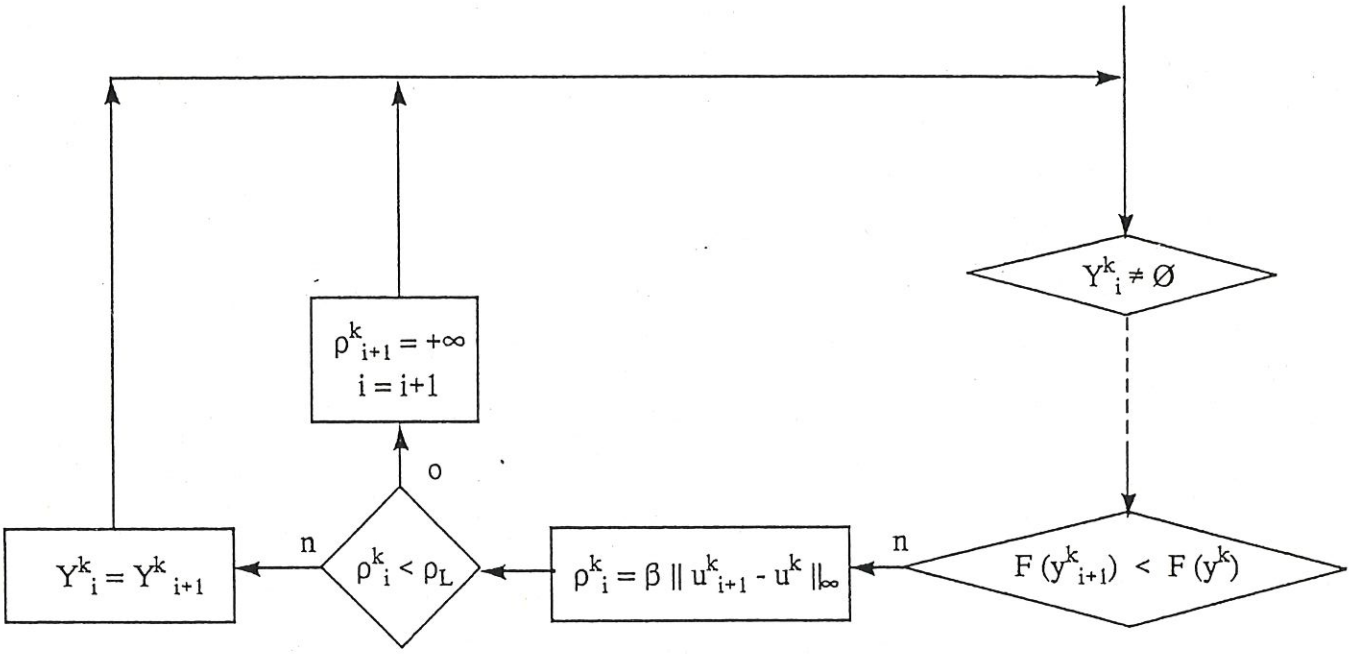
  $F(y^k_{i+1}) \geq F(y^k)$, c'est-à-dire

  $f(x^k_{i+1},y^k_{i+1}) \geq f(u^k)$ où x^k_{i+1} est la solution du problème $P(y^k_{i+1})$

  alors, on va diminuer le rayon de la région de confiance autour du point u^k, dans la problème (M^k_i). Cependant, il existe une borne inférieure ρ_L pour le rayon de la région de confiance. Si on atteint cette borne, pour améliorer la linéarisation de f et de g dans (M^k_i), on ajoute dans les contraintes de (M^k_i) l'approximation linéaire de f et de g au point $u^k_{i+1} = (x^k_{i+1},y^k_{i+1})$. On effectue donc une **linéarisation par morceaux** de f et de g.

  On obtient alors un nouveau problème (M^k_{i+1}) et on incrémente i d'une unité, tout en réinitialisant le rayon de la région de confiance à $+\infty$.

Sous forme d'organigramme, cela nous donne, pour le pas nul, le schéma suivant:



Pour terminer cette description de la méthode, voyons le processus d'initialisation de l'algorithme:

1) Initialisation des données:

$\rho_L > 0$ et $\rho_L \geq \rho_{min} = \min \{ \|z-y\|_\infty \mid y,z \in Y\}$

$\beta \in (0,1)$

$y^1 \in Y$

$Y^1 = Y$

$k = i = 1$

2) Recherche d'un point initial admissible pour (P_0):

a) on résout le problème non-linéaire continu $P(y^1)$

b) si $P(y^1)$ est admissible, on pose

$y^1_1 = y^1$

$\rho^1_1 = +\infty$,

$Y^1 = Y^1 \backslash \{y^1\}$

$Y^1_1 = Y^1$

$u^1 = (x^1,y^1)$ où x^1 est la solution du problème $P(y^1)$.

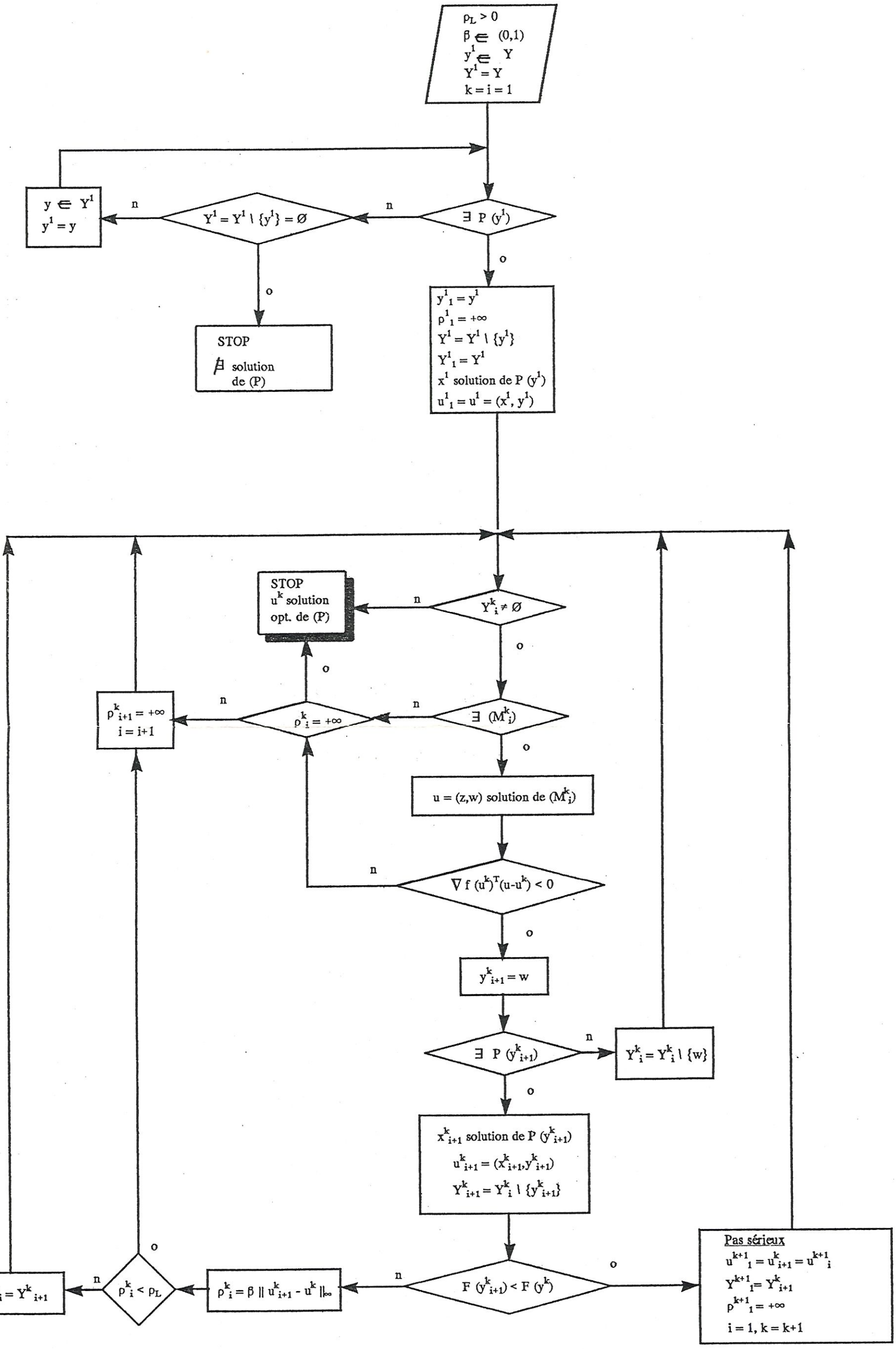
c) sinon, on pose $Y^1 = Y^1 \backslash \{y^1\}$.

Si $Y^1 = \emptyset$, STOP: la procédure se termine et il n'existe pas de solution de (P), vu que le théorème 2.2.1 nous a prouvé que les problèmes (P_0) et (P) sont équivalents.

Sinon, on choisit un nouveau point y dans Y^1 et on pose $y^1 = y$. Retour en (a).

Conclusion :

La méthode étudiée dans ce chapitre peut se représenter comme suit:


$\rho_L > 0$
$\beta \in (0,1)$
$y^1 \in Y$
$Y^1 = Y$
$k = i = 1$
$\exists\ P(y^1)$
n
$Y^1 = Y^1 \setminus \{y^1\} = \emptyset$
n
$y \in Y^1$
$y^1 = y$
o
STOP
$\nexists$ solution de (P)
o
$y^1_1 = y^1$
$\rho^1_1 = +\infty$
$Y^1 = Y^1 \setminus \{y^1\}$
$Y^1_1 = Y^1$
x^1 solution de $P(y^1)$
$u^1_1 = u^1 = (x^1, y^1)$
$Y^k_i \neq \emptyset$
n
STOP
u^k solution opt. de (P)
o
$\exists\ (M^k_i)$
n
$\rho^k_i = +\infty$
n
$\rho^k_{i+1} = +\infty$
$i = i+1$
o
$u = (z,w)$ solution de (M^k_i)
$\nabla f(u^k)^T(u-u^k) < 0$
n
o
$y^k_{i+1} = w$
$\exists\ P(y^k_{i+1})$
n
$Y^k_i = Y^k_i \setminus \{w\}$
o
x^k_{i+1} solution de $P(y^k_{i+1})$
$u^k_{i+1} = (x^k_{i+1}, y^k_{i+1})$
$Y^k_{i+1} = Y^k_i \setminus \{y^k_{i+1}\}$
$F(y^k_{i+1}) < F(y^k)$
n
o
Pas sérieux
$u^{k+1}_1 = u^k_{i+1} = u^{k+1}_i$
$Y^{k+1}_1 = Y^k_{i+1}$
$\rho^{k+1}_1 = +\infty$
$i = 1,\ k = k+1$
$\rho^k_i = \beta \| u^k_{i+1} - u^k \|_\infty$
$\rho^k_i < \rho_L$
n
o
$Y^k_i = Y^k_{i+1}$

Précisions :

- Si $Y^k_i = \emptyset$, tous les candidats possibles ont été considérés et c'est donc la dernière valeur admissible trouvée qui est le minimiseur de (P), c'est-à-dire u^k.

- Si le test $\nabla f(u^k)^T (u-u^k) < 0$ n'est pas satisfait, cela signifie que le point u^k est un minimiseur local ou global, ou bien un point isolé (cf section 2.4). Dès lors, le test " $\rho^k_i = +\infty$? " permet de voir s'il s'agit d'un minimiseur global ou non.

Remarques :

On ne décrira pas ici les méthodes employées pour résoudre les problèmes (M^k_i) et les problèmes $P(y^k)$.

2.4. Conditions d'optimalité et convergence :

Rappelons pour commencer les hypothèses qui sont faites sur f et g:

f et g sont deux fonctions différentiables sur $\mathbb{R}^n \times \mathbb{R}^m$ et convexes.

Les définitions de " minimiseur local " et de " point isolé " ont déjà été données dans l'introduction de ce travail.

Les théorèmes suivants vont justifier les trois tests d'arrêt utilisés dans l'algorithme:

Théorème 2.4.1. :

Soit u, solution du problème (M^k_i).

Si $\nabla f(u^k)^T (u - u^k) \geq 0$

Alors, u^k est un minimiseur local de (P) ou c'est un point admissible de (P) isolé.

preuve :

Supposons que u^k n'est pas un point isolé.

Soit u* un point admissible pour (P)

t.q. $\| u^* - u^k \| < \rho^k_i$

Comme f et g sont convexes, u* est admissible pour (M^k_i).

En effet, u* est admissible pour (P),

or,

le problème (P) peut encore s'exprimer comme

$$\begin{aligned} \min \quad & \mu \\ s.c. \quad & f(u) \leq \mu \\ & g(u) \leq 0 \\ & u \in X \times Y \\ & \mu \in \mathbb{R} \end{aligned}$$

Si on note $l_f(.)$ et $l_g(.)$ les linéarisations de f et g en un point quelconque, comme f et g sont convexes, on a:

$l_f(u^*) \leq f(u^*)$ et $l_g(u^*) \leq g(u^*)$

Donc,

- $l_f(u^*) \leq \mu$
- $l_g(u^*) \leq 0$
- $u^* \in X \times Y$
- $\| u^* - u^k \| < \rho^k_i$
- $\mu \in \mathbb{R}$

Si on note $\mu^* = l_f(u^*)$, on a que

$$f(u^*) \geq \mu^* \qquad (1)$$

Si μ est la solution optimale de (M^k_i), on a donc (2)

$$\begin{aligned} f(u^*) &\geq \mu^* && \text{(a)} \\ &\geq \mu && \text{(b)} \\ &= \max_{1 \leq j \leq i} \{f(u^k_j) + \nabla f(u^k_j)^T (u - u^k_j)\} && \text{(c)} \\ &\geq f(u^k_1) + \nabla f(u^k_1)^T (u - u^k_1) && \text{(d)} \\ &\geq f(u^k) \end{aligned}$$

car (a) : on a la relation (1)

preuve :

On définit:

$D \quad = \{u \in X \times Y \quad | \ u \text{ est admissible pour (P)}\}$

$D^k_i \quad = \{u \in X \times Y^k_i \quad | \ u \text{ est admissible pour (P)}\}$

$\forall\, u \in D^k_i$, on a :

$$f(u^k_j) + \nabla f(u^k_j)^T (u - u^k_j) \le f(u) \le \mu \qquad j = 1..i$$

$$g(u^k_j) + \nabla g(u^k_j)^T (u - u^k_j) \le g(u) \le 0 \qquad j = 1..i$$

car f et g sont convexes et que u est admissible pour (P)

Donc, comme (M^k_i) est non-admissible par hypothèse, on a nécessairement que

$$\| u - u^k_i \|_\infty > \rho^k_i$$

car c'est la seule contrainte de (M^k_i) qui puisse être mise en défaut.

Si $\forall\, u \in D \setminus D^k_i$ on a $\| u - u^k_i \|_\infty > \rho^k_i$,

alors, $\forall\, u \in D \cup D^k_i$ on a $\| u - u^k_i \|_\infty > \rho^k_i$

c'est-à-dire u^k est isolé.

Sinon, $\exists\, u \in D \setminus D^k_i$ t.q. $\| u - u^k_i \|_\infty \le \rho^k_i$.

Considérons $\bar{y}$, sa partie discrète. ($u = (\bar{x},\bar{y})$)

On doit avoir:

$\exists\, j < k, \exists l$ t.q. $\bar{y} = y^j_l$

En effet, normalement u est admissible pour (M^k_i),

mais $u \notin D^k_i$

donc, sa partie discrète a été enlevée de l'ensemble discret admissible avant le calcul de u^k.

Comme on n'augmente jamais la valeur de la fonction f dans l'algorithme, on doit avoir que

$$\min_{x \in X} f(x,\bar{y}) \geq f(u^k) \qquad (1)$$

Donc

$$f(u) \geq \min_{x \in X} f(x,\bar{y}) \qquad (a)$$

$$\geq f(u^k) \qquad (b)$$

car (a) : $u = (\bar{x},\bar{y})$ et par définition du minimum

(b) : par (1)

■

Corollaire 2.4.2.1. :

Si (M^k_i) est non-admissible avec $Y^k_i \neq \emptyset$ et $\rho^k_i = +\infty$,

Alors,

u^k est un minimiseur global de (P)

preuve :

évident car alors D^k_i doit être $= \emptyset$ dans le théorème 2.4.2 vu que $\|u - u^k_i\|_\infty \not> +\infty$

$\forall\, u \in X \times Y^k_i$

■

Théorème 2.4.3. :

Si $Y^k_i = \emptyset$

Alors, u^k est solution optimale de (P)

preuve :

Si $Y^k_i = \emptyset$, c'est que tous les points admissibles ont été envisagés, et comme, par construction de l'algorithme, la suite $(f(u^k))_j$ est une suite décroissante, on doit avoir que le dernier point retenu est optimal, c'est-à-dire u^k est optimal.

■

Théorème 2.4.4. :

L'algorithme atteint une solution en un nombre fini d'itérations.

preuve :

L'algorithme a trois possibilités d'arrêt:

Deux d'entre elles sont liées au problème (M^k_i) où $\rho^k_i = +\infty$:

1) si $\nabla f(u^k)^T (u - u^k) \geq 0$, alors, par le corollaire 2.4.1.1., u^k est solution globale de (P)
2) si (M^k_i) n'est pas admissible, alors, par le corollaire 2.4.2.1., u^k est minimiseur global de (P)

Enfin, on s'arrête au plus tard quand tous les points discrets admissibles ont été examinés. Comme par hypothèse, l'ensemble Y des points discrets admissibles est fini, la preuve est terminée.

■

Conclusion

Nous avons dans ce mémoire décrit deux méthodes de résolution , chacune se rapportant à un type bien précis de problèmes d'optimisation discrets.

Toutes deux se servent de linéarisations du problème de départ. Dans la première méthode, on construit itérativement un modèle linéaire par morceaux de la fonction objectif, restreinte à l'ensemble discret Y. La minimisation de ce modèle fournit le prochain point itératif.

Dans la seconde méthode, on linéarise la fonction objectif f et la contrainte g en un point, à chaque itération. Dans certains cas bien précis, on améliore le modèle en linéarisant f et g en plusieurs points.

Pour obtenir les modèles linéaires (par morceaux), on doit se servir de la notion de sous-gradient, qui est une généralisation pour les fonctions non-différentiables de la notion de gradient.

La façon de choisir le sous-gradient, dans l'algorithme du sous-gradient, est un gros problème auquel une grande partie du premier chapitre est consacrée. Ce choix est très important pour la vitesse de convergence de l'algorithme, et le résultat auquel on aboutit au terme du chapitre n'est pas satisfaisant à 100%, comme nous l'avons expliqué dans la conclusion 1.9. Cet aspect de la méthode peut donc encore être amélioré.

Remarquons que dans la seconde section, ce problème du choix du sous-gradient (lors de la détermination de $\nabla f(x^k,y^k)$ et de $\nabla g(x^k,y^k)$) ne se pose pas car les fonctions f et g sont supposées différentiables.

D'autre part, bien que la convergence de ces deux algorithmes soit assurée théoriquement, rien n'est dit en ce qui concerne la vitesse de convergence. L'analyse de ce problème est donc une voie possible pour des recherches ultérieures.

Enfin, aucun test numérique n'est disponible pour la deuxième méthode. Il serait donc intéressant, dans l'avenir, de voir le comportement de cet algorithme sur des problèmes concrets.

Références

[1] FICHEFET J., *Eléments de programmation linéaire*, Cours de première licence mathématique, Facultés Universitaires Notre-Dame de la Paix, Namur, 1982

[2] LOH et PAPALAMBROS, *A sequential linearization approach for solving mixed discrete nonlinear design optimization problems*, Technical Report UM-MEAM-89-08, Design Laboratory, The University of Michigan, Ann Arbor, 1989

[3] SHEIMBERG de MAKLER S., *A Benders algorithm for solving a mixed discrete mathematical programming problem*, Manuscrit non-publié, Facultés Universitaires Notre-Dame de la Paix, Namur, 1991

[4] STRODIOT J.J., *Notes du Cours de programmation non-linéaire,* deuxième licence mathématique, Facultés Universitaires Notre-Dame de la Paix, Namur, 1993

[5] YUAN X., *Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés*, Recherche opérationnelle, vol 22,n°4, p 331 à 346, 1988

[6] ZHIJUN WU, *A subgradient Algorithm for nonlinear integer programming and its parallel implementation*, Ph. D.,Rice University, HOUSTON, 1991